Title
From Fully-Supervised, Single-Task to Scarcely-Supervised, Multi-Task Deep Learning for Medical Image Analysis

Author
Imran, Abdullah-Al-Zubaer

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

From Fully-Supervised, Single-Task to Scarcely-Supervised, Multi-Task Deep Learning for Medical Image Analysis

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Abdullah-Al-Zubaer Imran

2020

ABSTRACT OF THE DISSERTATION

From Fully-Supervised, Single-Task to Scarcely-Supervised, Multi-Task Deep Learning for Medical Image Analysis

by

Abdullah-Al-Zubaer Imran
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2020
Professor Demetri Terzopoulos, Chair

Image analysis based on machine learning has gained prominence with the advent of deep learning, particularly in medical imaging. To be effective in addressing challenging image analysis tasks, however, conventional deep neural networks require large corpora of annotated training data, which are unfortunately scarce in the medical domain, thus often rendering fully-supervised learning strategies ineffective.

This thesis devises for use in a variety of medical image analysis applications a series of novel deep learning methods, ranging from fully-supervised, single-task learning to scarcely-supervised, multi-task learning that makes efficient use of annotated training data. Specifically, its main contributions include (1) fully-supervised, single-task learning for the segmentation of pulmonary lobes from chest CT scans and the analysis of scoliosis from spine X-ray images; (2) supervised, single-task, domain-generalized pulmonary segmentation in chest X-ray images and retinal vasculature segmentation in fundoscopic images; (3) largely-unsupervised, multiple-task learning via deep generative modeling for the joint synthesis and classification of medical image data; and (4) partly-supervised, multiple-task learning for the combined segmentation and classification of chest and spine X-ray images.

The dissertation of Abdullah-Al-Zubaer Imran is approved.

Guy Van den Broeck

Kai-Wei Chang

Song-Chun Zhu

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2020

To my mother and father

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I cannot thank Professor Demetri Terzopoulos enough for supporting me as my advisor throughout my PhD program. Long before coming to UCLA, I got to know about his revolutionary work on deformable models for image segmentation, particularly the snake model. While I was pursuing a master's degree at Delaware State University in medical image analysis, I contacted him inquiring about a potential PhD position in his group. During my SPIE 2016 conference trip in California, I got a chance to meet him in person at UCLA and thereafter was determined to pursue my PhD under his supervision. Without his support and assistance, I would not be where I am today. Throughout my PhD studies, he provided his unconditional support whenever I needed it, be it research matters or academic issues. He gave me a clear research direction and often spent hours discussing the progress of my research, even within his extremely tight schedule. Without his guidance and support, this thesis would never have eventuated. I am and will always remain grateful to him.

I would like to express my appreciation to Professors Song-Chun Zhu, Guy Van den Broeck, and Kai-Wei Chang for serving on my thesis committee and offering me their kind advice to make my dissertation stronger. Thanks to my committee, I was able to build my knowledge and skills to make progress in my research and write this dissertation.

Furthermore, I would like to express my deep appreciation to previous advisors Professor David Pokrajac at Delaware State University and Professor Predrag Bakic at the University of Pennsylvania. I thank them immensely for all their valuable inputs to my career, especially for guiding me through an excellent medical imaging research setting. I would also like to express my gratitude to my bachelor thesis advisor, Professor Boshir Ahmed at the Rajshahi University of Engineering & Technology, for piquing my interest in image processing research.

Special thanks goes to my industry mentors and colleagues, Dr. Nima Tajbakhsh at VoxelCloud, Inc., Dr. Kunal Vaidya, Dr. Alvin Chen, and Dr. Ameet Jain at Philips Corporate Research, and Dr. Zhen Qian and Dr. Chao Huang at Tencent Medical AI, for

giving me the opportunity to engage in research activities in corporate settings.

I am grateful to all the coauthors of my publications related to this dissertation for their valuable contributions to my work.

I am very fortunate to have been surrounded by wonderful labmates in the UCLA Computer Graphics and Vision Laboratory who helped me progress through my PhD program. I would like to thank Dr. Tomer Weiss, Dr. Masaki Nakada, Dr. Tao Zhao, Dr. Garett Ridge, Dr. Arul Jeyraj, Mr. Ali Hatamizadeh, and Mr. Alan Litteneker.

During the first year of my PhD program, I was funded by a research assistantship made possible by an unrestricted gift from VoxelCloud, Inc., which I gratefully acknowledge and for which I thank Dr. Xiaowei Ding.

Lastly, I would like to thank my parents, Mozammel Biswas and Firoza Khatun, who raised me with unconditional love. Without their support I could not have achieved anything significant in my life, let alone come this far. I also thank my sister Shumaya Khatun, and my friends Al Arafat and Mijanur Rahman Sahed for continuously encouraging me in my work.

VITA

2012	B.S. Computer Science & Engineering Rajshahi University of Engineering & Technology (RUET) Rajshahi, Bangladesh
2012–2013	Lecturer of Computer Science & Engineering Northern University Dhaka, Bangladesh
2013–2014	Lecturer of Computer Science & Engineering Ahsanullah University of Science & Technology Dhaka, Bangladesh
2016	M.S. Computer Science Delaware State University Dover, Delaware, USA
2017	Lecturer of Electrical & Computer Engineering North South University Dhaka, Bangladesh
2017–2018	Research Assistant Computer Graphics & Vision Laboratory University of California, Los Angeles Los Angeles, California, USA
2018–2020	Teaching Assistant Computer Science Department University of California, Los Angeles Los Angeles, California, USA

2018	Research & Development Intern
	Philips Corporate Research
	Boston, Massachusetts, USA
2019	Research Intern
	Tencent Medical AI
	Palo Alto, California, USA

CHAPTER 1

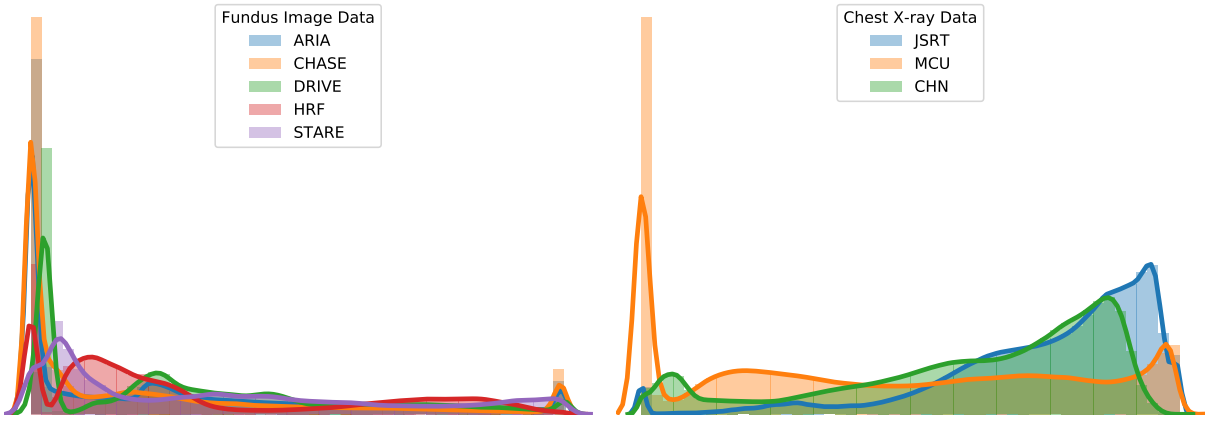
Introduction

A general objective of medical image computing is to make predictions. In the machine learning approach, predictions are enabled based on models trained on collections of medical image data. In deep learning, which involves neural network models with numerous hidden layers, significant hierarchical relationships within the data can be discovered algorithmically, replacing the traditional laborious hand-crafting of image feature extractions. However, training deep neural networks usually requires copious data. Given set of properly annotated or labeled training data (input-output pairs), a supervised learning algorithm can learn the mapping from input to output. A well-trained supervised model can make predictions about previously unseen examples.

1.1 The Problem of Limited Training Data

Although there has been explosive progress in the production of vast quantities of high resolution images, large collections of properly labeled or annotated images required for fully-supervised learning remain scarce. Expert annotation of medical images is expensive, time-consuming, and prone to human subjectivity, inconsistency, and error. Even when properly labeled datasets become available, they are often limited in size due to privacy issues and are highly imbalanced and non-uniformly distributed. In an imbalanced dataset, there will be an over-representation of common medical problems and an under-representation of rarer conditions. Such biases make the training of neural networks across multiple classes with consistent effectiveness very challenging.

The success of deep learning is based on the assumption that the training data are



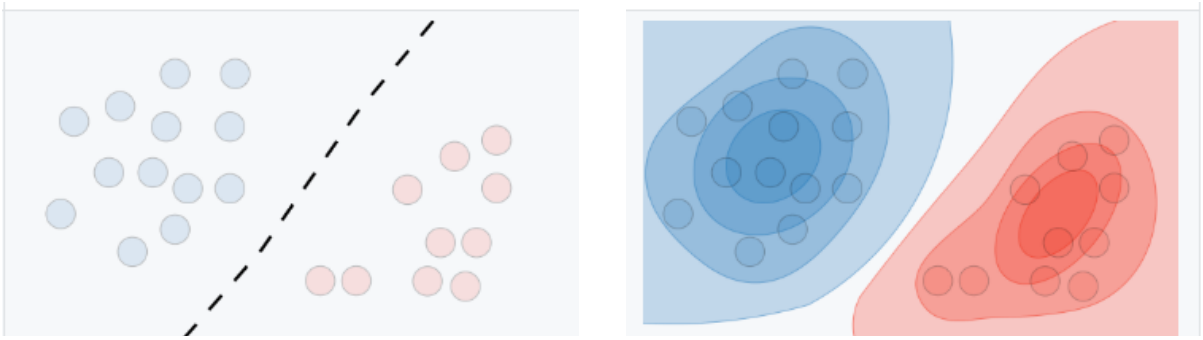
(a) (b)

Figure 1.1: Disparities in the distributions of different datasets with varying data sources, sizes, ratios of normal/abnormal cases, etc.

independently and identically distributed (i.i.d.). However, this assumption may not hold in real world scenarios. Moreover, the problem is exacerbated due to varying medical conditions, imaging configurations and modalities, among other factors. Generalization refers to how well a model performs on previously unseen data. Difficulties further compound if the unseen data have different distributions than the training data (Figure 1.1). This leads to the domain shift problem—when a model trained on data from one source does not generalize well to out of distribution (o.o.d.) test data from a different source, which can cause even the most sophisticated deep learning models to make non-intuitive, erroneous predictions.

1.2 Generative Models

The limited training data problem is traditionally mitigated through simplistic and often cumbersome data augmentation schemes, usually by creating new training examples through translation, rotation, flipping, etc. The missing or mismatched label problem may be addressed by evaluating similarity measures over the training examples; however, this is not always robust and its effectiveness depends largely on the performance of the similarity measuring algorithms. A more intuitive approach could be via generative modeling where the aforementioned issues are tackled algorithmically and automatically.



(a) Discriminative (b) Generative

Figure 1.2: Illustration of how a a generative model works compared to a discriminative model. The discriminative model learns only a decision boundary—a conditional distribution $p(y \mid x)$—whereas the generative model learns a joint probability distribution $p(x, y)$, thus enabling both generation and discrimination. (Images from https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning.)

Unlike discriminative models that directly learn a classifier to make predictions given the training data, generative models learn the data generation process along with a classifier (Ng and Jordan, 2002). Figure 1.2 illustrates the working principle of a generative model compared to a discriminative model. Generative models offer a powerful way to estimate data distributions through unsupervised learning, by constructing density over all the observable quantities (MacKay, 2003). This property of generative models is useful for unsupervised learning in realistic image generation and the clustering of images. Deep learning-based generative models work mostly based on latent coding, which helps elucidate hidden phenomena and similarities among observations. More generally, treating generative modeling as an auxiliary task leads to semantically meaningful, unsupervised representation learning. Furthermore, generative modeling can infer causal relations as opposed to mere data correlations (Rottman and Hastie, 2014).

Our medical image analysis models are expected to learn from small quantities of labeled data, while also leveraging larger quantities of unlabeled examples, and to achieve effective generalization for consistent performance across different data domains. To this end, it is preferable to employ generative models over their discriminative counterparts.

1.3 Semi-Supervised Learning

Especially in the medical imaging domain, there is growing interest in leveraging potentially large quantities of unlabeled data along with limited quantities of labeled data. Such approaches are called semi-supervised learning (SSL). For them to be effective, however, the knowledge gained from the unlabeled medical image data must be significant to the model (Chapelle et al., 2009). Depending on how unlabeled data are leveraged, semi-supervised learning can be accomplished in several ways, and this has recently emerged as a growing body of research, yielding schemes such as transfer learning, domain adaptation, self-supervised learning, adversarial learning, and multitask learning.

In transfer learning, a model is first trained on similar tasks in some other domains with labeled data, and the pre-trained model is then fine-tuned with a limited set of labeled data in the target domain. In domain adaptation, a model is trained on labeled data from the source domain and unlabeled examples from the target domain, and then evaluated on unseen examples from the target domain. Self-supervised learning is closely related to transfer learning. Unlike transfer learning, the model is pre-trained on some surrogate tasks in the same domain, and then the pre-trained model is evaluated on the actual medical image analysis tasks. Self-supervised learning is usually based on the assumption that the predicted labels from the original data and the augmented data should be the same. Adversarial learning augments the class labels with an additional label to differentiate the generated data and real data. A well balanced adversarial learning helps towards learning useful visual features from the unlabeled data. Multitask learning (MTL) is basically defined as optimizing more than one loss within the same model. In MTL, multiple related tasks are jointly learned, which results in better generalization of the model.

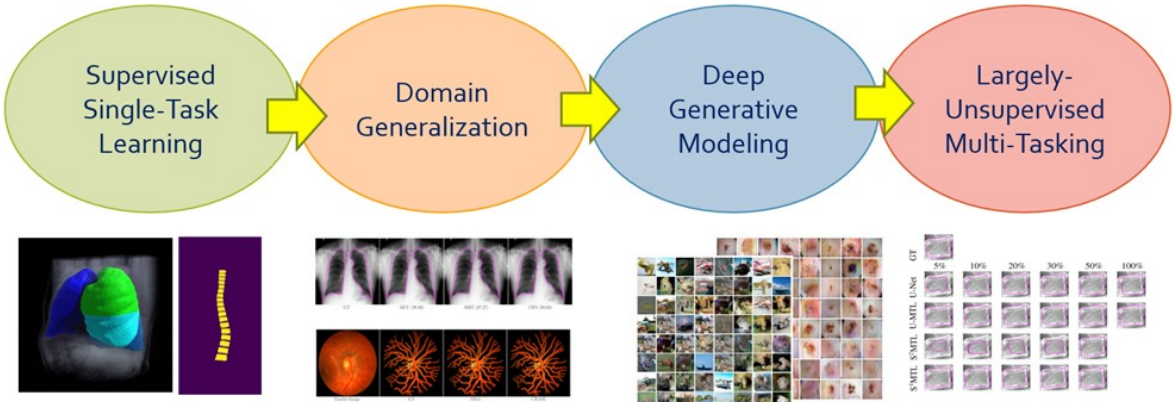


Figure 1.3: Spectrum of thesis contributions. Our contributions progress from supervised, single-task learning to sparingly-supervised multi-task learning.

1.4 Contributions

As illustrated in Figure 1.3, the contributions reported in this thesis range from fully-supervised, single-task to sparingly-supervised, multi-task deep learning, through domain generalization and deep generative modeling. In greater detail, the six contributions are as follows:

1. **Volumetric medical image segmentation:** Automatic, reliable lung lobe segmentation is crucial to the diagnosis, assessment, and quantification of pulmonary diseases. Existing pulmonary lobe segmentation techniques are prohibitively slow, undesirably rely on prior (airway/vessel) segmentation, and/or require user interactions for optimal results. To address the need for accurate and robust lobe segmentation, we have pursued a fully automatic and reliable deep learning solution based on a Progressive Dense V-Network (PDV-Net). Our 3D PDV-Net model inputs an entire CT volume and generates accurate segmentation of the lung lobes in about 2 seconds in only a single forward pass of the network, eliminating the need for any user interaction or any prior segmentation of the lungs, vessels, or airways, which are common assumptions in the design of existing models. An extensive robustness analysis of our method demonstrates reliable lobe segmentation of both healthy and pathological lungs in CT images acquired by scanners from different vendors, across various CT scan protocols and acquisition parameters. This work is

published in (Imran et al., 2018, 2019a).

2. **2D medical image segmentation & disease quantification:** Scoliosis is a congenital disease in which the spine is deformed from its normal shape. Measurement of scoliosis requires labeling and identification of vertebrae in the spine. Spine radiographs are the most cost-effective and accessible modality for imaging the spine. Reliable and accurate vertebrae segmentation in spine radiographs is crucial in image-guided spinal assessment, disease diagnosis, and treatment planning. Conventional assessments rely on tedious and time-consuming manual measurement, which is subject to inter-observer variability. A fully automatic method that can accurately identify and segment the associated vertebrae is unavailable in the literature. Leveraging a carefully-adjusted U-Net model with progressive side outputs, we propose an end-to-end segmentation model that provides a fully automatic and reliable segmentation of the vertebrae associated with scoliosis measurement. Our experimental results from a set of anterior-posterior spine X-Ray images indicate that our model, which achieves an average Dice score of 0.993, promises to be an effective tool in the identification and labeling of spinal vertebrae, eventually helping doctors in the reliable estimation of scoliosis. Moreover, estimation of Cobb angles from the segmented vertebrae further demonstrates the effectiveness of our model. This work is published in (Imran et al., 2019c, 2020a).

3. **Deep generative modeling for simultaneous image generation and classification:** From relatively small corpora of training data, deep generative models can learn to generate realistic images approximating real-world distributions. In particular, the proper training of Generative Adversarial Networks (GANs) and Variational AutoEncoders (VAEs) enables them to perform semi-supervised image classification. Combining the power of these two models, we introduce Multi-Adversarial Variational autoEncoder Networks (MAVENs), a novel deep generative modeling that incorporates an ensemble of discriminators in a VAE-GAN network in order to perform simultaneous adversarial learning and variational inference. We apply

MAVENs to the generation of synthetic images and propose a new distribution measure to quantify the quality of these images. Our experimental results with only 10% labeled training data from the computer vision and medical imaging domains demonstrate performance competitive to state-of-the-art semi-supervised models in simultaneous image generation and classification tasks. This work is published in (Imran and Terzopoulos, 2019a, 2021).

4. **Generalized and improved medical image segmentation:** The performance of fully-supervised models for various image analysis tasks (e.g., anatomy or lesion segmentation from medical images) is limited to the availability of massive amounts of labeled data. Given small sample sizes, such models are prohibitively data biased with large domain shift. To tackle this problem, we propose a novel end-to-end medical image segmentation model, namely Progressive Adversarial Semantic Segmentation (PASS), which can make improved segmentation prediction without requiring any domain-specific data during training time. Our extensive experimentation with 8 public diabetic retinopathy and chest X-ray image datasets, confirms the effectiveness of PASS for accurate vascular and pulmonary segmentation, both for in-domain and cross-domain evaluations. This work appears in (Imran and Terzopoulos, 2020).

5. **Semi-supervised multi-task learning:** We propose a novel multi-task learning model for jointly learning a classifier and a segmentor, from chest X-ray images, through semi-supervised learning. In addition, we propose a new loss function that combines absolute KL divergence with Tversky loss (KLTV) to yield faster convergence and better segmentation performance. Based on our experimental results using a novel segmentation model, an Adversarial Pyramid Progressive Attention U-Net (APPAU-Net), we hypothesize that KLTV can be more effective in improving generalizability of multi-task learning models while being competitive in segmentation-only tasks. This work is published in (Imran and Terzopoulos, 2019b).

6. **Self-supervised, semi-supervised, multi-task learning:** Leveraging adversarial training and self-supervision, we propose a novel general purpose semi-supervised, multiple-task model—namely, self-supervised, semi-supervised, multi-task learning (S^4MTL)—for accomplishing two important tasks in medical imaging, segmentation and diagnostic classification. Experimental results on chest and spine X-ray image datasets suggest that our S^4MTL model significantly outperforms semi-supervised single task, semi/fully-supervised multi-task, and fully-supervised single-task models, even with a 50% reduction of class and segmentation labels. We hypothesize that our proposed model can be effective in tackling limited annotation problems for joint training, not only in medical imaging domains, but also for general-purpose vision tasks. This work is published in (Imran et al., 2020c).

1.5 Overview

The remainder of this dissertation is organized as follows:

Chapter 2 briefly reviews the developments in supervised learning, semi-supervised learning, deep generative modeling, self-supervision, multitasking, and domain generalization, especially applied to medical image segmentation, image synthesis, and image classification. We also review related work on applications such as pulmonary lobe segmentation from chest CT, measurement of scoliosis from spine radiographs, simultaneous image generation and classification, semi-supervised multitasking for combined image classification and segmentation, and improved segmentation with domain generalization without domain-specific data.

Chapter 3 develops the deep learning models and associated algorithms, ranging from fully-supervised, single-task learning to scarcely-supervised, multi-task learning, which were described in the previous section and Figure 1.3.

Chapter 4 presents an extensive collection of experimental results with our proposed algorithms and models. Appendix A describes all the datasets used, data insights, class distributions, and dataset partitionings for training, testing, and validation.

Chapter 5 concludes the dissertation with a summary of our work and promising future research directions.

CHAPTER 2

Related Work

We will now review prior work relevant to the six contributions listed in the previous chapter. The material in this chapter is organized into two parts; first we review supervised learning approaches to medical image segmentation followed by approaches requiring only limited supervision.

2.1 Supervised Learning for Medical Image Segmentation

Image segmentation can classically be defined as the partitioning of an image into non-overlapping, coherent regions that are homogeneous based on some characteristic such as intensity or texture (Gonzalez et al., 2004). Image segmentation is usually considered the most important part of medical image analysis as it extracts regions of interest (ROI) and simplifies the representation by focusing attention to smaller regions crucial to disease diagnosis/prognosis. Segmentation simply assigns labels to a set of constituent regions, whereas semantic segmentation understands and recognizes image regions at the pixel level; i.e., semantic segmentation assigns class labels to each pixel in an image. In this section, we review related work on fully-supervised learning approaches semantic segmentation in general and prior deep learning methods for the segmentation of anatomical structures from 2D (e.g., X-rays) and 3D (e.g., CT scans) medical image data.

With the advent of deep convolutional neural networks (CNNs) in computer vision and medical image analysis, automatic feature learning algorithms via deep learning emerged for medical image segmentation. These methods perform segmentation via pixel-wise classification, overcoming the limitations of the conventional pixel or super-pixel based

methods requiring hand-crafted features (Mehra and Neeru, 2016). CNNs employed in medical image segmentation can be categorized into two approaches: patch-wise and whole image-based processing. Ciresan et al. (2012) proposed a patch-wise sliding window-based convolutional network for the automatic segmentation of neuronal membranes in electron microscopy images. Kamnitsas et al. (2017) proposed a dual pathway 3D CNN architecture with a fully connected random field (CRF) for refining patch-based brain lesion segmentation from multi-channel MRI patient data. Long et al. (2014) proposed semantic segmentation using an end-to-end fully convolutional network (FCN) by transforming classification networks (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015). This opened up the pixel-wise classification over the full images differing the earlier patch-wise, sliding window strategies. The fully-connected layers from the classification networks were replaced by convolution layers which allowed the model perform dense prediction (pixel-to-pixel) on arbitrary sized inputs and outputs. Moreover, coarse, high layer information were combined with fine, low layer information which resulted in predicting finer details.

Ronneberger et al. (2015a) proposed the U-Net architecture for biomedical image segmentation which won the ISBI 2015 cell tracking challenge. The architecture employs a contraction path for capturing context and a symmetric expansion path for the precise localization of object(s) of interest. At every layer of the expansion path, high resolution features from the contraction layer are combined in order to reconstruct it and achieve better segmentation prediction. Gu et al. (2019) proposed a context encoder network (CE-Net) that captures more high-level information while preserving spatial information, and applied the model to several 2D medical image segmentation applications (vessel detection, lung segmentation, etc.).

Extending the previous U-Net architecture, Çiçek et al. (2016) proposed the 3D U-Net for volumetric segmentation of Xenopus kidney. Replacing the 2D operations by their 3D counterparts, the 3D U-Net model was applied in semi-automated and automated setup for dense prediction in volumetric images. (Milletari et al., 2016) proposed a 3D CNN model namely V-Net for the segmentation of prostate volumes in MR images. To optimize

the training, a Dice-based objective function was proposed which deals with the imbalance between the number of foreground and background voxels. Each of the compression and decompression stages learns a residual function that ensures faster convergence.

By extracting intra-slice features with a 2D DenseU-Net and hierarchically aggregating volumetric contexts with a 3D DenseU-Net, Li et al. (2018) proposed a hybrid H-DenseUNet for liver and tumor segmentation from CT volumes.

Among different variants of the U-Nets and V-Net, attention U-Net (Oktay et al., 2018), nested U-Net (Zhou et al., 2019), hierarchical 3D U-Net (Roth et al., 2017) and 3D dense V-Net for abdominal segmentation(Gibson et al., 2018) are noteworthy.

2.1.1 3D Segmentation

Our work on 3D segmentation (Imran et al., 2018, 2019a), which is developed in Section 3.1.1, focuses on the segmentation of pulmonary lobes in 3D CT images of the chest. We will next discuss the lung lobe segmentation problem and review the relevant literature.

Human lungs are divided into five lobes. The inner membrane of the lung (visceral pleura) folds towards the center of the lung and creates double layer fissures that define the five lobes. The lobar boundaries are made of two major (oblique) fissures and a minor (horizontal) fissure. As shown in Figure 2.1, the left lung has two lobes separated by a major fissure—the upper (superior) lobe and the lower (inferior) lobe. Along with upper and lower lobes, the right lung has a middle lobe; a minor fissure separates the upper lobe from the middle lobe and a major fissure separates the lower lobe from the middle lobe. Each of the five lobes is functionally independent, with its own bronchial and vascular systems.

Automatic segmentation of the lung lobes is important for both clinical and technical purposes. From the clinical perspective, automatic lung lobe segmentation can help radiologists review chest CT scans more efficiently. This is because radiologists often report their pulmonary findings by indicating the affected lung lobe, whose identification

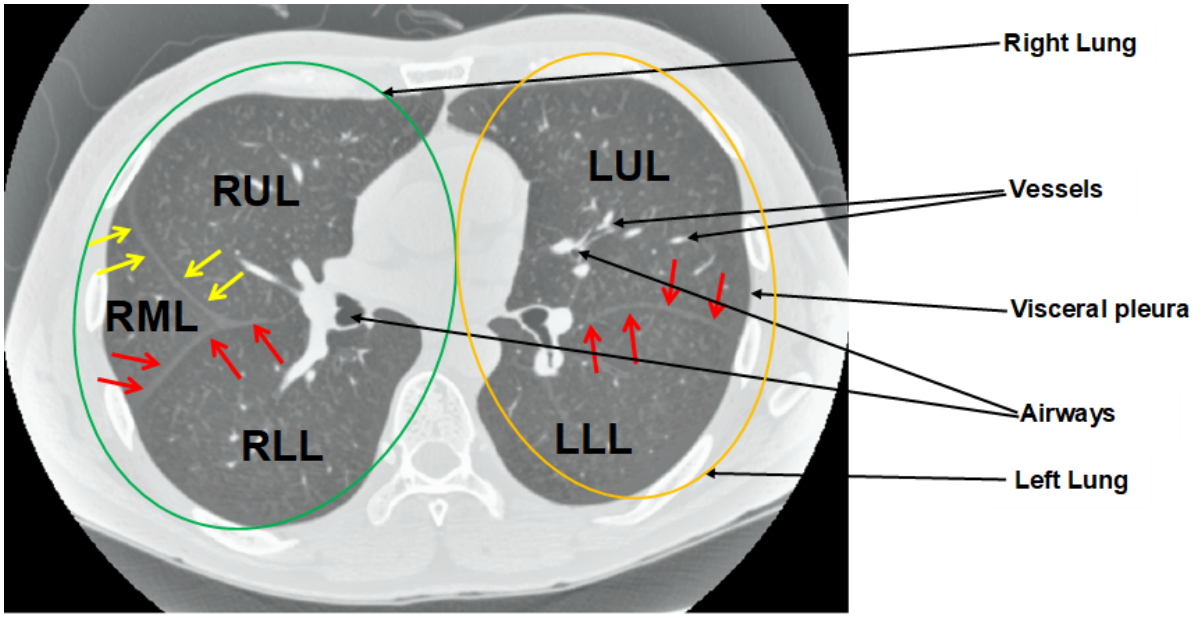


Figure 2.1: An axial lung CT slice with visible fissures. In the left lung, the left upper lobe (LUL) and left lower lobe (LLL) are defined by a major fissure (indicated by red arrows). In the right lung, the right upper lobe (RUL), right middle lobe (RML), and right lower lobe (RLL) are defined by a major fissure (indicated by red arrows) and a minor fissure (indicated by yellow arrows).

requires them to navigate through the nearby slices and search for fissure lines, which are often visually indistinct. Automatic lung lobe segmentation can eliminate the need for such a tedious and time-consuming process. From the technical perspective, accurate lung lobe segmentation can assist several subsequent clinical tasks, including nodule malignancy prediction (cancers mostly occur in the left or right upper lobes), automatic lobe-aware report generation for each nodule (see Figure 2.2), and assessment and quantification of chronic obstructive pulmonary diseases (COPD) and interstitial lung diseases (ILD), by narrowing down the search space to the lung lobes most-likely to be affected.

However, identifying fissures poses a challenge for both human and machine perception. First, fissures are most often incomplete, not extending to the lobar boundaries. This is shown in Figure 2.2 where the horizontal fissure is incomplete, unlike the oblique fissures. Several studies in the literature have confirmed the incompleteness of fissures as a very common phenomenon. After reviewing 100 fixed and inflated lung specimens, Raasch et al. (1982) found incomplete right major fissures in 70% of the cases, left major in 46%

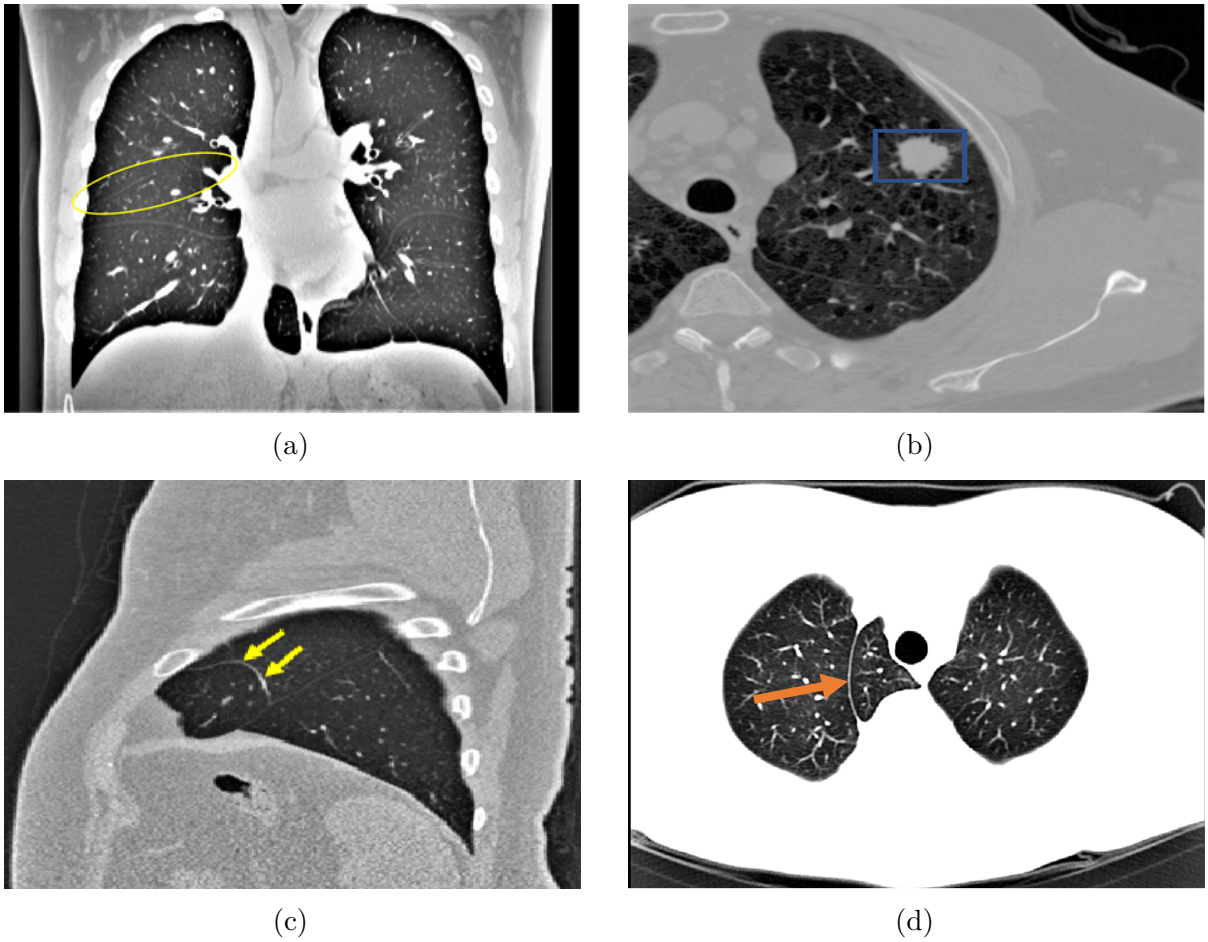


Figure 2.2: (a) A coronal slice where the major fissures are complete and visible, but the minor fissure (circled) is incomplete. (b) Nodule shown in the bounding box. (An example nodule report: `5mm nodule found in the left upper lobe`). (c) Accessory fissure (arrows) in a left lung sagittal slice, which looks similar in shape to a minor fissure. (d) Azygos fissure (arrow) in an axial slice creates an extra lobe (azygos lobe) in the right lung.

of the cases, and 94% across the minor fissures. Moreover, the studies of Gulsun et al. (2006) and Aziz et al. (2004) also showed more than 50% incompleteness in pulmonary fissures. Second, the visual characteristics of lobar boundaries change in the presence of pathologies. The changes could also be related to their thicknesses, locations, and shapes. Third, there also exist other fissures in the lungs that can be misinterpreted as the major and minor fissures that separate the lobes. Examples include accessory fissures (sagittal slice in Figure 2.2) and azygos fissures (axial slice in Figure 2.2).

There have been several efforts to segment lung lobes using semi-automatic and

automatic techniques. We categorize these approaches into two groups: *reliant approaches*, which rely on a prior segmentation or anatomical information, and *non-reliant approaches*, which do not rely on such prior segmentations.

2.1.1.1 Prior-Based Segmentation

Reliant approaches require as input a segmentation mask of lungs or lobes (different modalities), airways and vessels, or fissure initialization. A good example of the latter is the work by Doel et al. (2012), in which lobe segmentation is performed based on an initialization via fissure detection. In another example of fissure initialization,Iwano et al. (2013) proposed semi-automatic and automatic lobe segmentation methods based on region-growing. The semi-automatic approach requires major and minor fissure initialization, whereas for the automatic approach, recognition of lobar bronchi and localization of fissures are performed prior to the final lobar segmentation. On average, the semi-automatic approach takes approximately 80 seconds and the automatic approach takes approximately 44 seconds per case.

A number of works depend on prior segmentation of airways, vessels, or fissures. The work by Bragman et al. (2017) is a good representative, wherein the method relies on the prior segmentation of airways and vessels. Specifically, a population model of fissure priors was constructed and combined with patient-specific anatomical information for non-parametric surface fitting. Despite the promising results, the model lacks robustness and its reliance on prior knowledge limited the study. In recent work, Giuliani et al. (2018) proposed an approach to segment lobes from an approximate segmentation based on the airway tree. The final lobe segmentation was generated by combining the approximate segmentation with all the lung structures (airways, vessels, lungs, and fissures) segmentation using a multilevel graph cut algorithm. This segmentation method is highly reliant on the quality of the prior airway and vessel segmentations, as well as anatomical knowledge. Lassen and van Rikxoort (2013) proposed a watershed-based lobe segmentation method by combining anatomical information from lungs, fissures, vessels, and bronchi. Despite

reporting improved segmentation in the presence of incomplete fissures, the failure of individual prior segmentations limited the performance of the overall segmentation. Based on this work, Lassen-Schmidt et al. (2017) proposed an interactive lobe segmentation method to interactively correct lobe segmentation error through user inputs. However, this improvement was obtained at the price of prolonged segmentation sessions. Lim et al. (2016) performed quantification of emphysema in 66 patients with moderate to severe emphysema who had undergone CT for lung volume reduction planning. They used lobar segmentation from four different prototypes for inter-software variability in lobe-wise emphysema quantifications. Although the lobe segmentation performance is not reported, it is dependent on prior airway and vessel segmentation.

Other works also rely on prior lung or lobe segmentation masks. For example, Bauer et al. (2018) segmented the lung lobes in the expiration phase based on a prior lobe segmentation mask obtained from a CT image acquired in the inspiration phase. An automated lung and lobe segmentation pipeline was proposed by Blaffert et al. (2010), in which a lung model mesh based on watershed segmentation is adapted to lobar segmentation. Final lobe regions are obtained by adjusting based on overlaid lungs in a post-processing step. However, the authors do not report a quantitative evaluation of lobar segmentation. The model takes 20 seconds to perform lobar segmentation in each CT scan.

2.1.1.2 Atlas-Based Segmentation

Another variation of reliant segmentation is registration using mutual information with a previously segmented atlas. The performance of final lobe segmentation is greatly dependent on the performance of the segmentation algorithm used in creating a reference atlas. Among atlas-based approaches for lobe segmentation, Ross et al. (2010) employed the thin-plate spline and a maximum a posteriori estimation method using a manually-defined atlas as a reference. Fissure points were selected based on the atlas and the final lobe segmentation was generated after a post-processing step. Although this method

does not rely on any prior airway and vessel segmentation, the execution time was long. Moreover, the creation of the atlas is very cumbersome and prone to poor results in pathological lung cases. By contrast, Pu et al. (2009) performed lobe segmentation by fitting an implicit function to fissures without reliance on prior airway or vessel segmentation. Although they achieved good accuracy for healthy lungs, the performance of their method degraded in the case of lungs with abnormal orientations. Unlike the other atlas-based segmentations, van Rikxoort et al. (2010) made use of multiple atlases for lobe segmentation. Their method showed promise albeit at the expense of slow execution.

2.1.1.3 Non-Reliant Segmentation

Recently, a few convolutional neural-network-based lobe segmentation techniques have been proposed (George et al., 2017; Ferreira et al., 2018; Wang et al., 2018). The segmentation method of George et al. (2017) employs a 2D fully convolutional network followed by a 3D random walker algorithm. This approach does not rely on a prior segmentation of airways or vessels nor on any pre-computed atlases; however, it cannot generate lobe segmentation in a single pass, nor in an end-to-end manner. Furthermore, the 3D random walker algorithm relies on a number of heuristics for the initialization of seeds and weights. Ferreira et al. (2018) proposed a lobe segmentation model based on a fully regularized V-Net model with deep supervision and carefully chosen regularization. Although the performance looks impressive, the model was trained with few examples, so it lacks generalizability and may not be effective for varying CT scan cases. A 3D Dense Net-based lobe segmentation method was proposed by Wang et al. (2018). Although they reported good accuracy for pathological lungs, their lobe segmentation method relies on prior lung segmentation and assumes the presence of five lobes, which might not always be the case (e.g., (LOLA11, 2011)).

Our work (Imran et al., 2018, 2019a), developed in the next chapter, mitigates the aforementioned limitations—namely, reliance on prior masks, slow runtime, and lack of robustness—through an end-to-end learning network. Without relying on any prior

airway/vessel segmentation or anatomical knowledge or atlases, our method performs lobe segmentation in a single pass of the network. Owing to the full utilization of the 3D context in our model, the resulting lobe segmentation is smooth and nearly noise-free, which eliminates the need for any subsequent post-processing to fill holes or remove noisy patches from outside the lung area. Our method shows promise for the potential clinical use in quantification of pulmonary diseases and automatic generation of radiological reports.

2.1.2 2D Segmentation

In this section, we review the literature relevant to our work on the segmentation of vertebrae in 2D spinal X-ray images and the development of a segmentation-based pipeline for the measurement and analysis of scoliosis (Imran et al., 2019c, 2020a), which is reported in Section 3.1.2.

Scoliosis is an abnormal condition defined by spinal curvature towards the left or right. Early detection is key and, when accurate, it can lead to better treatment planning (Weinstein et al., 2008). Radiography (X-Ray) is the preferred imaging technique for clinical analysis and measurement of scoliosis as it is highly available, inexpensive, and yields quick results. Conventional spine image analysis tasks involve tedious manual labor with hand-crafted feature extraction for the measurement of scoliosis. The Cobb angle, the standard metric of scoliosis, is estimated by calculating the angle between the two tangents of the upper and lower end plates of the upper and lower vertebrae. A person with a 10° or greater Cobb angle is usually considered for scoliosis diagnosis (Kim et al., 2010). Figure 2.3 illustrates the procedure for the calculation of the Cobb angle through the labeling of relevant vertebrae in an X-Ray image.

Conventionally, measurement and assessment, which requires the identification and labeling of specific vertebral structures, is manually performed by clinicians. However, the manual measurement of scoliosis faces several difficulties. First, large anatomical variation between patients and low tissue contrast in spinal X-Ray images make it challenging to

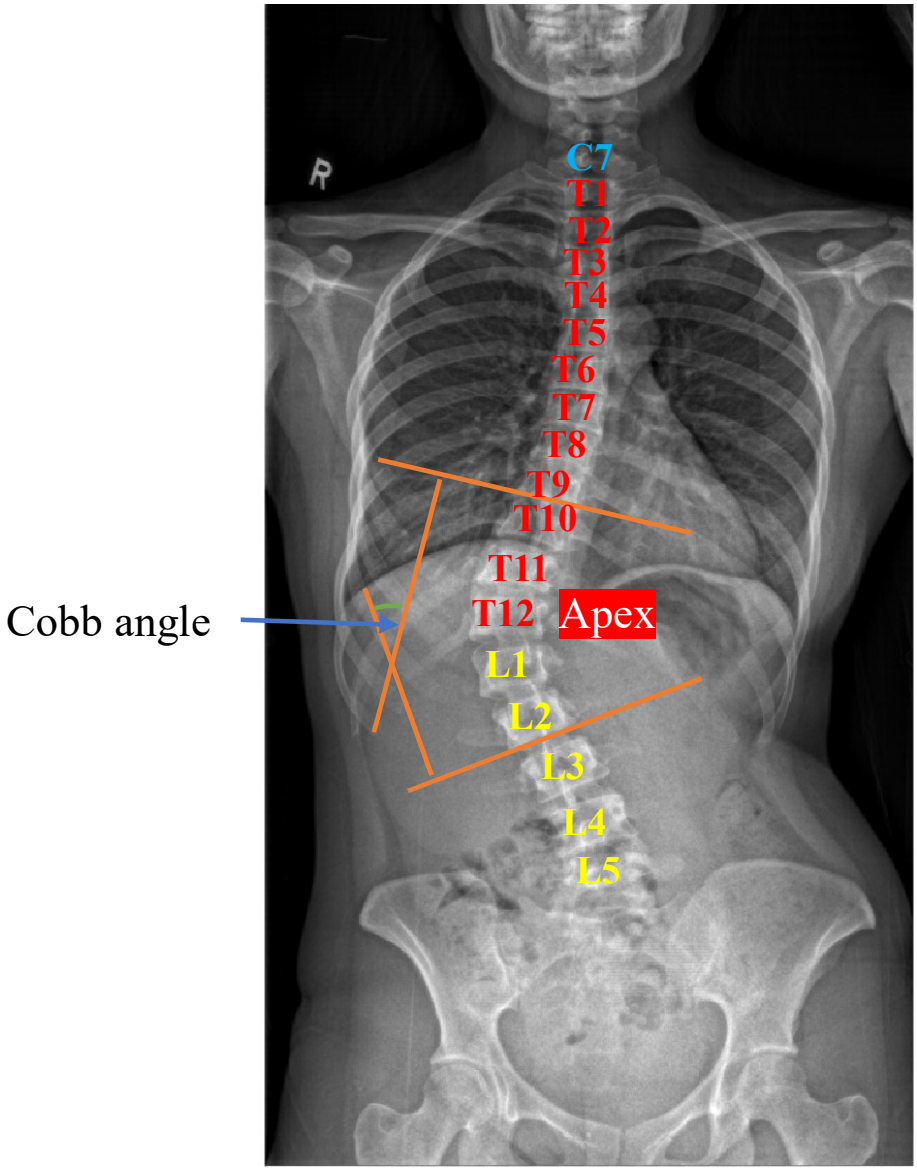


Figure 2.3: Illustration of the Cobb angle calculation in an anterior-posterior spine X-ray, by selecting the most tilted upper vertebra above the apex and the most tilted lower vertebra below the apex. From the extended upper edge of the upper vertebra and lower edge of the lower vertebra, tangents are drawn and the intersection angle is calculated as the Cobb angle.

accurately and reliably assess the severity of scoliosis (Wu et al., 2017), and effects on the spine and body as a whole, as well as on individual vertebra, pose extra difficulty in the quantification of scoliosis (Kawchuk and McArthur, 1997). Second, measurement error is prevalent in the routine clinical assessment of scoliosis due to instrumentation, vertebral rotation, and patient positioning (Kim et al., 2010), and 5°–10° intra- or greater inter-observer variation has commonly been reported in measuring the Cobb angle (Beauchamp et al., 1993; Pruijs et al., 1994).

While several methods for vertebrae segmentation and scoliosis measurement are available, this approach is still under-explored in the literature. Existing vertebrae segmentation methods rely on manual interaction (Mateusiak and Mikolajczyk, 2019), hand-crafted feature engineering limited to customized parameters (Taghizadeh et al., 2019; Anitha and Prabhu, 2012), follow patch-based approaches that lose full spatial context (Qadri et al., 2019; Horng et al., 2019), are limited in scope and fail to consider all the required vertebrae at a time (Lessmann et al., 2019), etc. For Cobb angle estimation, a minimum bounding rectangle was used for the patch-wise segmented vertebrae (Horng et al., 2019), an approach that relies on pre-processing steps including spinal region isolation and vertebrae detection. Kusuma (2017) proposed a K-means and curve-fitting approach for Cobb angle measurement that requires a set of pre-processing steps (Kusuma, 2017). Other Cobb angle estimation methods have been proposed based on directly finding vertebrae corners as a form of regression task (Wu et al., 2018; Sun et al., 2017; Imran et al., 2019b; Wu et al., 2017). Although promising, these supervised methods are less viable for clinical applications because of low accuracy, due to the loss of fine details in the process, and the lack of explainability.

2.2 Limited Supervision

The scarcity of labeled or annotated medical image data and/or access to substantial quantities of unlabeled data motivates efforts to train deep learning models with limited supervision. We review the literature on limited supervision, including semi-supervised

learning facilitated by deep generative modeling, adversarial learning, self-supervision, multi-task learning, and domain generalization techniques, in the following sections.

2.2.1 Semi-Supervised Learning

Semi-supervised learning has recently been explored both in computer vision and medical imaging due to the availability of vast amounts of unlabeled data and computing power to process them. Semi-supervised learning is usually performed with a small portion of labeled and a larger portion of unlabeled data, assuming that both are from the same or similar distributions. In standard protocols, semi-supervised models are evaluated by retaining only a portion of the labels from a dataset while the remainder are treated as unlabeled data (Zhai et al., 2019). Depending on the approach to gaining information from large quantities of unlabeled data, semi-supervised learning can be performed in at least two different ways—self-supervised learning and adversarial training.

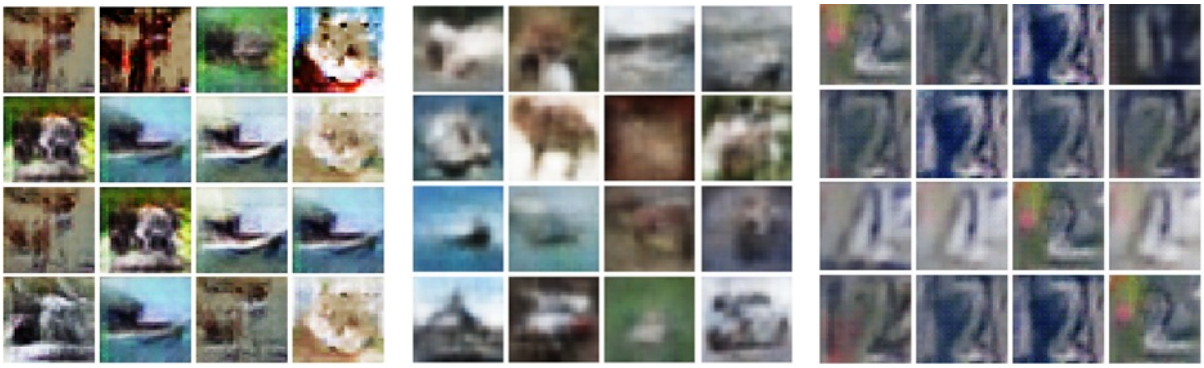
Self-supervised learning is similar to unsupervised learning in its goal of using a vast amount of unlabeled data to learn visual representation without any human annotation. Usually, self-supervised learning is performed by formulating a pretext or surrogate task only on the unsupervised data portion. Examples of pretext tasks could include image reconstruction, image colorization, predicting image rotations, etc. In self-supervision, the data itself lends to supervision; i.e., proxy labels created from the data on which training can provide useful visual features from unlabeled data. Tajbakhsh et al. (2019a) showed the effectiveness of training models from pre-trained surrogate tasks in different medical imaging applications, including diabetic retinopathy classification, nodule detection, and lung lobe segmentation with limited labeled data. Moreover, without training separately, both the pretext and downstream tasks can be combined in jointly learning useful visual features. Tran (2019) proposed a semi-supervised learning scheme based on self-supervised regularization, where the model is trained like full-supervision—a supervised branch for the labeled data and a self-supervised branch for the unlabeled data—to predict some geometric transformations.

Adversarial learning is closely related to the Generative Adversarial Networks (GANs) Goodfellow et al. (2014). In adversarial learning, class labels are augmented with an additional label to distinguish generated data from real data. A well balanced generator-discriminator helps towards learning useful visual features from the unlabeled data Donahue et al. (2016). Adversarial learning can effectively be adapted to semi-supervised learning for classification of both natural and medical images (Salehinejad et al., 2018; Imran and Terzopoulos, 2019a). Adversarial learning has also been utilized in segmentation (semantic-aware generative adversarial nets (Chen et al., 2018), structure correcting adversarial nets (Dai et al., 2018), etc.) as well as in disease classification (semi-supervised domain adaptation (Madani et al., 2018), attention-guided CNN (Guan et al., 2018)).

2.2.2 Deep Generative Modeling

With the advent of deep generative models such as Variational AutoEncoders (VAEs) (Kingma and Welling, 2013) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), the ability to learn underlying data distributions from training samples has become practical in common scenarios where there is an abundance of unlabeled data. With minimal annotation, efficient semi-supervised learning could be the preferred approach (Imran and Terzopoulos, 2019b). More specifically, based on small quantities of annotation, realistic new training images may be generated by models that have learned real-world data distributions (Figure 2.4a). Both VAEs and GANs may be employed for this purpose.

VAEs can learn dimensionality-reduced representations of training data and, with an explicit density estimation, can generate new samples. Although VAEs can perform fast variational inference, VAE-generated samples are usually blurry (Figure 2.4b). On the other hand, despite their successes in generating images and semi-supervised classifications, GAN frameworks remain difficult to train and there are challenges in using GAN models, such as non-convergence due to unstable training, diminished gradient issues, overfitting, sensitivity to hyper-parameters, and mode collapsed image generation (Figure 2.4c).



(a) Good (b) Blurry (c) Mode collapsed

Figure 2.4: Image generation based on the CIFAR-10 dataset (Krizhevsky and Hinton, 2009): (a) Relatively good images generated by a GAN. (b) Blurry images generated by a VAE. Based on the SVHN dataset (Netzer et al., 2011): (c) Mode collapsed images generated by a GAN.

Several techniques have been proposed to stabilize GAN training and avoid mode collapse. Nguyen et al. (2017) proposed a model where a single generator is used alongside dual discriminators. Durugkar et al. (2016) proposed a model with a single generator and feedback aggregated over several discriminators, considering either the average loss over all discriminators or only the discriminator with the maximum loss in relation to the generator's output. Neyshabur et al. (2017) proposed a framework in which a single generator simultaneously trains against an array of discriminators, each of which operates on a different low-dimensional projection of the data. Mordido et al. (2018), arguing that all the previous approaches restrict the discriminator's architecture thereby compromising extensibility, proposed the Dropout-GAN, where a single generator is trained against a dynamically changing ensemble of discriminators. However, there is a risk of dropping out all the discriminators. Feature matching and minibatch discrimination techniques have been proposed (Salimans et al., 2016) for eliminating mode collapse and preventing overfitting in GAN training.

Realistic image generation helps address problems due to the scarcity of labeled data. Various architectures of GANs and their variants have been applied in ongoing efforts to improve the accuracy and effectiveness of image classification. The GAN framework has been utilized as a generic approach to generating realistic training images that synthetically

augment datasets in order to combat overfitting; e.g., for synthetic data augmentation in liver lesions (Frid-Adar et al., 2018), retinal fundi (Guibas et al., 2017), histopathology (Hou et al., 2017), and chest X-rays (Salehinejad et al., 2018; Imran and Terzopoulos, 2019b). Calimeri et al. (2017) employed a LAPGAN (Denton et al., 2015) and Han et al. (2018) used a WGAN (Arjovsky et al., 2017) to generate synthetic brain MR images. Bermudez et al. (2018) used a DCGAN (Radford et al., 2015) to generate 2D brain MR images followed by an autoencoder for image denoising. Chuquicusma et al. (2018) utilized a DCGAN to generate lung nodules and then conducted a Turing test to evaluate the quality of the generated samples. GAN frameworks have also been shown to improve the accuracy of image classification via the generation of new synthetic training images. Frid-Adar et al. (2018) used a DCGAN and an ACGAN (Odena et al., 2017) to generate images of three liver lesion classes to synthetically augment the limited dataset and improve the performance of a convolutional neural net (CNN) in liver lesion classification. Similarly, Salehinejad et al. (2018) employed a DCGAN to artificially simulate pathology across five classes of chest X-rays in order to augment the original imbalanced dataset and improve the performance of a CNN in chest pathology classification.

The GAN framework has also been utilized in semi-supervised learning architectures to leverage unlabeled data alongside limited labeled data. The following efforts demonstrate how incorporating unlabeled data in the GAN framework has led to significant improvements in the accuracy of image-level classification: Madani et al. (2018) used an order of magnitude less labeled data with a DCGAN in semi-supervised learning yet showed comparable performance to a traditional supervised CNN classifier and furthermore demonstrated reduced domain over-fitting by simply supplying unlabeled test domain images. Springenberg (2015) combined a WGAN and CatGAN (Wang and Zhang, 2017) for unsupervised and semi-supervised learning of feature representation of dermoscopy images.

Despite the aforecited successes, GAN frameworks remain challenging to train, as we discussed above. Our MAVEN framework (Imran and Terzopoulos, 2019a, 2021), which we develop in Section 3.2.1, mitigates the difficulties of training GANs by enabling training

on a limited quantity of labeled data, preventing overfitting to a specific data domain source, and preventing mode collapse, while supporting multiclass image classification.

2.2.3 Domain Generalization

The domain shift problem is usually mitigated using two approaches: unsupervised domain adaptation and domain generalization. Domain generalization involves acquiring knowledge from an arbitrary number of related domains and applying it to previously unseen domains. In unsupervised domain adaptation, a model is pre-trained on similar tasks in some other domain(s) with labeled data, and the pre-trained model is then fine-tuned with a limited set of labeled data in the target domain. Domain adaption can also be performed to learn a generic representation where the model is fully-supervised for source data and unsupervised for data from the target domain (Yang et al., 2019; Zhuang et al., 2019). These methods typically rely on the availability of unlabeled (Chen et al., 2019; Huo et al., 2018) or even labeled data (Zhang et al., 2018b; Dou et al., 2018) for the target domain. Therefore, with the unavailability of labeled/unlabeled data from the target domains or disparate data distributions, such models become less useful.

Also, approaches are described in the literature for learning generalized albeit error-prone initial segmentation across domains and applying different techniques to improve the final segmentation. For example, post-processing approaches inconsistently improve segmentation results but require extensive parameter tuning (Kamnitsas et al., 2017) and error propagation before the post-processing (Larrazabal et al., 2019). ErrorNet is another method proposed recently that can learn error correction in a systematic manner via learning a prior distribution of the segmentation masks (Tajbakhsh et al., 2019b). However, all of these methods are based on explicit error propagation whether it is handcrafted or model-derived. This adds vulnerability to the models for segmentation prediction and might increase shifts across different application tasks. Moreover, the over-reliance on the post-processing or secondary error correction steps diminishes the quality of visual representation learning.

Departing from all the previous methods, we proposed the PASS model (Imran and Terzopoulos, 2020), which is developed in Section 3.2.2. PASS is end-to-end and fully automatic, devoid of any pre- or post-processing and explicit error designs (hand-crafted/systematic), and, most importantly, better generalized for tackling domain shifts in segmentation tasks.

2.2.4 Multitask Learning

Multitask learning (MTL) can be accomplished in several ways, such as learning from auxiliary tasks to support the main task (Liebel and Körner, 2018), learning to learn multitask models (Zhang et al., 2018a), joint learning of multiple tasks (Liu et al., 2019; Imran and Terzopoulos, 2019b), etc. By performing multiple tasks, the domain-specific information in the training signals of related tasks is actually improved (Caruana, 1993). MTL is particularly useful for implicit data augmentation through better representation, focusing attention on the most relevant features, learning one task through another, and regularizing among them (Ruder, 2017).

The literature includes several efforts on performing multiple tasks within the same model. Several prior efforts address multitask learning with CNNs and generative modeling. Rezaei et al. (2018) combined a set of auto-encoders with an LSTM unit and an FCN as discriminator for semantic segmentation and disease prediction. Girard et al. (2019) used a U-Net-like architecture coupled with graph propagation to jointly segment and classify retinal vessels. Mehta et al. (2018) proposed a Y-Net, with parallel discriminative and convolutional modularity, for the joint segmentation and classification of breast biopsy images. Another multitasking model was proposed by Yang et al. (2017) for skin lesion segmentation and melanoma-seborrheic keratosis classification, using GoogleNet extended to three branches for segmentation and two classification predictions. Khosravan and Bagci (2018) used a semi-supervised multitask model for the joint learning of false positive reduction and nodule segmentation from 3D computed tomography (CT) images.

We proposed novel SSL models for the joint classification and segmentation in an

adversarial learning framework. These include our APPAU-Net model (Imran and Terzopoulos, 2019c), which is developed in Section 3.2.3, and our S^{4}MTL model (Imran et al., 2020b), which is developed in Section 3.2.4 and further incorporates self-supervision for the unlabeled data.

CHAPTER 3

Models and Associated Learning Algorithms

This chapter develops our novel models and associated learning algorithms, first of the fully-supervised type and then of the semi-supervised type.

3.1 Fully-Supervised Learning Models

We target our supervised learning models to the tasks of fully automated pulmonary lobe segmentation in 3D CT 3D images of the chest and vertebra segmentation in anterior-posterior spine X-Ray images for the measurement of scoliosis.

3.1.1 3D Pulmonary Segmentation in Chest CT Images

3.1.1.1 Progressive Dense V-Net

Combining ideas from dense V-Networks (Gibson et al., 2018) and progressive holistically-nested networks (Harrison et al., 2017), we propose a new architecture—the Progressive Dense V-Network (PDV-Net), an end-to-end solution for organ segmentation in 3D volumetric data.

As shown in Figure 3.1, the input to the network is first down-sampled and concatenated with a strided $5 \times 5 \times 5$ convolution of the input with 24 kernels. The concatenation result is then passed to 3 dense feature blocks, each consisting of 5, 10, and 10 densely-wired convolution layers respectively. The growth rates of the dense blocks are set to 4, 8, and 16 respectively. All the convolutional layers in a dense block have a kernel size of $3 \times 3 \times 3$ and are followed by batch normalization and parametric rectified linear units

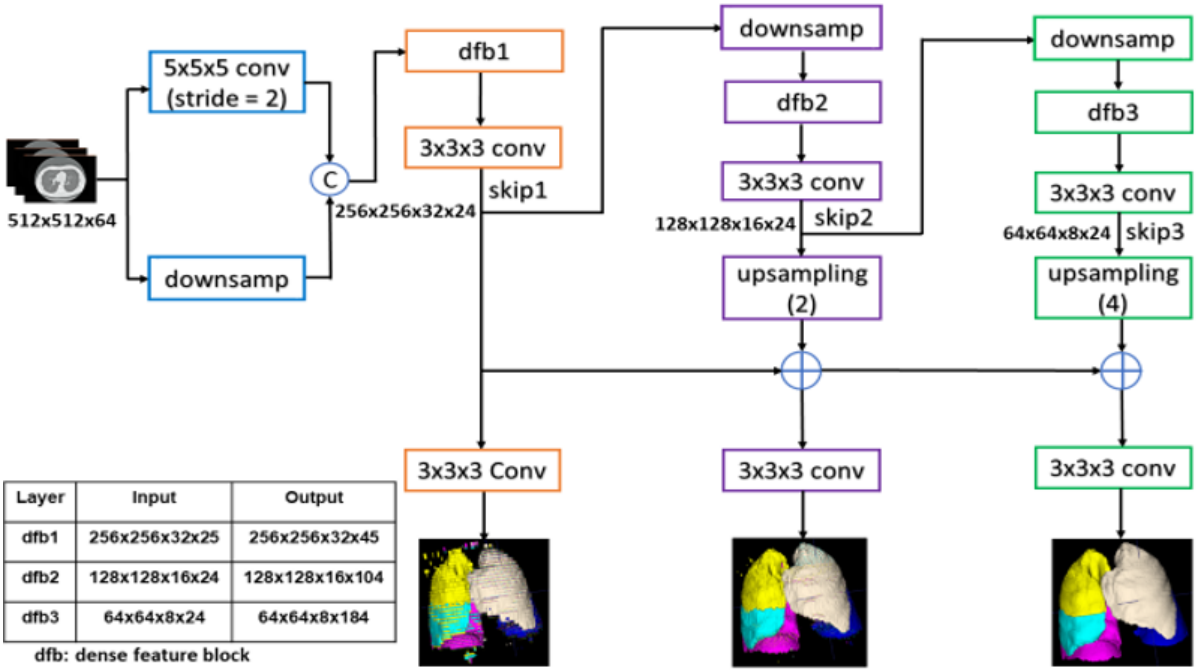


Figure 3.1: PDV-Net model for the segmentation of lung lobes. Segmentation outputs in different pathways are progressively improved to yield the final result.

(PReLU). The outputs of the dense feature blocks are consecutively utilized in low and high resolution passes via convolutional down-sampling and skip connections. This enables the generation of feature maps at three different resolutions. The outputs of the skip connections of the second and third dense feature blocks are further up-sampled in order to be consistent with the size of the output in the first skip connection. The feature maps from skip1 are passed to a convolutional layer followed by a softmax, which outputs the probability maps. In the second pathway, the feature maps from skip1 and skip2 are merged and the output probability maps are produced by a convolutional layer followed by softmax. Similarly, we obtain the final segmentation from the merged feature maps resulting from the skip2 and skip3 connections. Unlike the dense V-Net, the PDV-Net generates the final output by progressively improving the outputs at previous pathways.

The PDV-Net is trained using a subset S of a volumetric medical image dataset $\mathcal{D}$. The training set S contains 3D CT scan images and their corresponding ground truth labels. So, $S = (\mathcal{X}_n, \mathcal{Y}_n)$, for $n = 1, \ldots, N$, where the input CT volumes $\mathcal{X}_n^{(m)} = x_i^{(n)}; i = 1, \ldots |\mathcal{X}|_n$, and the corresponding ground truth labeled volumes $\mathcal{Y}_n^{(m)} = y_i^{(n)}; i = 1, \ldots |\mathcal{Y}|_n, y_l^{(n)} \in$

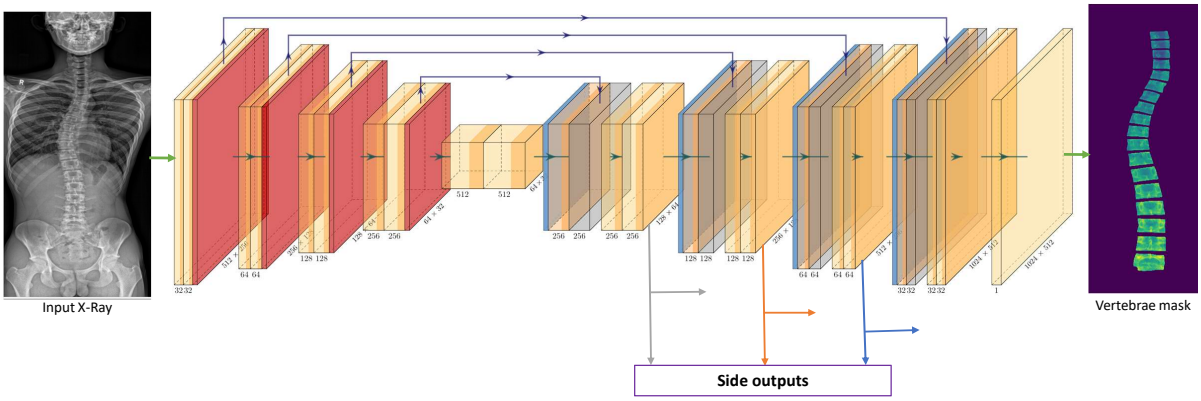


Figure 3.2: Architecture of our segmentation network (Progressive U-Net): Side outputs at three different stages of the decoder are generated and progressively added to the next stage side-output. The output from the third side-output is added to the last stage before the final convolution to generate the final segmentation output.

$\{0 \dots L\}$. Here, $|S|$ is the total number of training examples passed to the network and L is the number of labels provided in the ground truth data through per-voxel labeling (l). To train the PDV-Net, we use a Dice loss function (Milletari et al., 2016) at each level of the progressive network, which directly maximizes the similarity between the predicted values and the ground truth over all voxels. This loss properly handles the class imbalance problem prevalent in lung lobe segmentation: lung lobes have different sizes and background regions can be large. We employ a multi-class Dice for the segmentation task:

$$d = \sum_{l=1}^{L} \frac{\sum_{j=1}^{Z} p_j^l g_j^l}{\sum_{j=1}^{Z} (p_j^l)^2 + \sum_{j=1}^{Z} (g_j^l)^2}, \tag{3.1}$$

where Z is the total number of voxels, L is the number of classes, p_j^l denotes the predicted probabilities for each class, and g_j^l denotes the corresponding ground truth for each class.

3.1.2 Segmentation and Scoliosis Quantification in Spine Radiographs

3.1.2.1 Vertebrae Segmentation and Labeling

We perform binary segmentation of the spine with a well-distinguishable number n of vertebrae relevant to scoliosis analysis. To formulate the problem, we assume an unknown data distribution $p(X, Y)$ over images X and vertebrae segmentation labels Y. The model

Algorithm 1: Training for vertebrae segmentation in spine X-Ray images

Require:

Training data $x, y \in \mathcal{D}$ including spine X-Ray images x and reference vertebra segmentation masks y

Model architecture $\mathcal{F}_\phi$ with learnable parameters ϕ

Ensure:

for each step over $\mathcal{D}$ **do**

Sample minibatch $\mathcal{M} : x_{(i)} \sim p_{\mathcal{D}_{(x)}}$

Compute model outputs for the minibatch:
$\hat{y}_{(i)} \leftarrow \mathcal{F}_{(\phi)}(x)$

Calculate loss $\mathcal{L}_{(y,\hat{y})}$ for the model predictions

Update the model $\mathcal{F}$ along its gradient

$$\nabla_{\psi_\mathcal{F}} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \left[\mathcal{L}_{\mathcal{F}_{\left(y_{(i)}, \hat{y}_{(i)}\right)}} \right]$$

end for

has access to the labeled training set $\mathcal{D}_{(x,y)}$ sampled i.i.d. from $p(X, Y)$. As shown in Algorithm 1, the segmentation prediction network $\mathcal{F}_\phi$ is trained with a set of learnable parameters ϕ. We specify the objective as $\min_{\phi_\mathcal{F}} \mathcal{L}_{(y,\hat{y})}$, where y is the reference vertebrae mask and $\hat{y}$ is the model prediction in each of the training iterations.

Following the progressive dense V-Net model (Imran et al., 2018, 2019a), we propose a progressive U-Net with some careful adjustments in the U-Net (Ronneberger et al., 2015b). As shown in Figure 3.2, our model has an encoder and a decoder with skip connections. In each encoder layer, two 3×3 convolutions are followed by instance normalization, ReLU activation, and a 2×2 max-pooling. A dropout is applied in every encoder and decoder stage of the network. We generate side-outputs in every stage of the decoder. Progressively adding one side-output to the next, the segmentation performance is improved compared to collecting the final output from the final decoder stage in a U-Net. However, one key difference with (Imran et al., 2018) is that our model is trained without side-supervision. Only the side-outputs are generated and added progressively, yielding an improved segmentation at the final output. A convolution operation is performed to generate the side-output from each decoder stage. The progressive side-outputs also ensure

Algorithm 2: Cobb angle calculation.

Input: Vertebra mask $\hat{y}$
Output: Cobb angle θ

From the predicted mask $\hat{y}$, get all the contours
for each contour in contours **do**
  if Number of pixels $< a$ **then**
    //to remove any noisy patches if there are any
    Remove contour
  end if
end for
//This will give n contours of well-separated vertebrae
Extract four corner points from the vertebra contours
Order the extracted $4n$ corners from bottom to top
for up $= 1$ to $n - 2$ **do**
  for low $=$ up $+ 2$ to n **do**
    //with vertebra gap of 2
    Determine slope of the upper edge of the upper vertebra (m_{up})
    Determine slope of the lower edge of the lower vertebra (m_{low}).
    Calculate Cobb angle, $\tau_{(\text{up, low})} = \left|\tan^{-1}\left(\frac{m_{\text{up}} - m_{\text{low}}}{1 + m_{\text{up}} m_{\text{low}}}\right)\right|$
  end for
end for
Final Cobb angle, $\theta = \max(\tau)$

that micro-structure is not lost from any level of the decoder through the convolutional operations. We generate side outputs at $x/8$, $x/4$, and $x/2$ resolutions before the final output at x resolution. Therefore, the side output at resolution $x/8$ is added to the next decoder stage, and so on.

3.1.2.2 Measurement of Scoliosis

Our pipeline makes use of the vertebrae segmentation in estimating Cobb angles. Algorithm 2 automatically calculates the Cobb angle by analyzing the contours from the segmented mask. When well-separated from others, each of the contours represents a vertebra relevant to the measurement of scoliosis. To verify if a contour is actually associated to a relevant vertebra, we impose a minimum size on the number of contour pixels (a). After the extraction and ordering of the $4n$ corners, the most tilted upper

Table 3.1: Clinically accepted classification and treatment planning for adolescent scoliosis based on measured Cobb angles

Cobb Angle θ	Severity	Treatment Recommendation
$\theta < 10°$	normal	—
$10° < \theta < 25°$	mild	Check in every 2 years
$25° < \theta < 45°$	moderate	Wear a brace for 16–23 hours/day
$45° < \theta$	severe	Revision surgery in 20–30 years

vertebra and the most tilted lower vertebra are determined from the n relevant vertebrae. With a minimum vertebra gap of 2, Cobb angles are calculated for every combination of upper and lower vertebrae. From the slopes of the upper edge of the upper vertebra and the lower edge of the lower vertebra, Cobb angles ($\tau_{(up,low)}$) are calculated. Then the maximum τ is taken as the final Cobb angle θ.

Moreover, the severity of scoliosis can be categorized and appropriate treatment planning is performed depending on the calculated Cobb angle from the spine X-Ray of a patient. In our pipeline, we therefore perform an automatic diagnostic classification following the clinically recognized scoliosis severity classes, as shown in Table 3.1. Active treatment is typically not needed when it is mild and rigid braces can stop the progression of scoliosis when it is in moderate stage. Surgery is the last resort for severe cases, but it can be delayed for the adolescent period (Yang et al., 2016).

3.2 Learning From Limited Labeled Data

In the context of learning from limited labeled images, we next develop the following methods: (1) deep generative modeling for simultaneous image generation and classification, (2) domain generalization without domain specific data for improved medical image segmentation, (3) semi-supervised multi-task learning in an adversarial learning framework for joint image classification and segmentation, and (4) self-supervised semi-supervised multi-task learning for combined classification and segmentation of medical images.

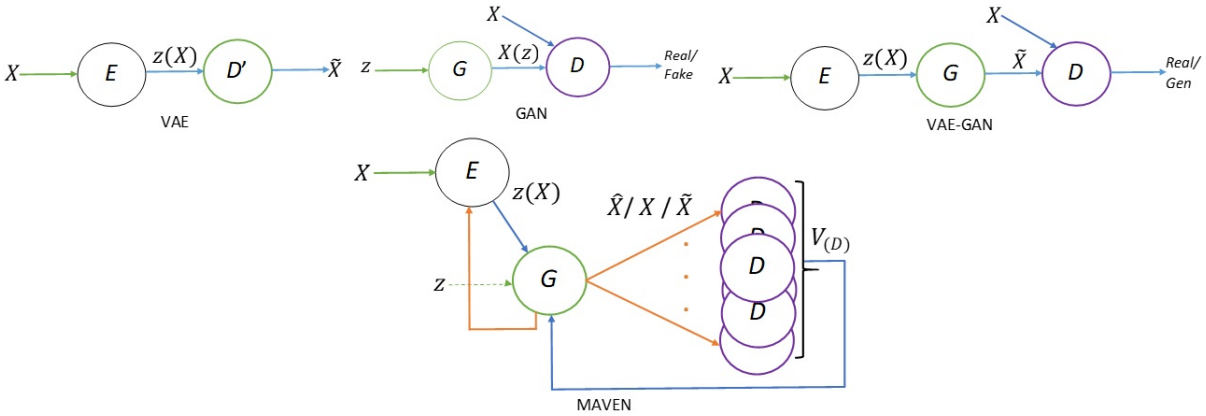


Figure 3.3: Our MAVEN architecture compared to those of the VAE, GAN, and VAE-GAN. In the MAVEN, inputs to D can be real data X, or generated data $\hat{X}$ or $\tilde{X}$. An ensemble ensures the combined feedback from the discriminators to the generator.

3.2.1 Simultaneous Image Generation and Classification

We develop a novel deep generative model namely, Multi-Adversarial Variation AutoEncoder Networks (MAVENs) for simultaneously performing realistic image generation and improved classification.

3.2.1.1 MAVENs

Figure 3.3 illustrates the models that serve as precursors to our MAVEN architecture.

The VAE is an explicit generative model that uses two neural nets, an encoder E and decoder D'. Network E learns an efficient compression of real data x into a lower dimensional latent representation space $z(x)$; i.e., $q_\lambda(z|x)$. With neural network likelihoods, computing the gradient becomes intractable; however, via differentiable, non-centered re-parameterization, sampling is performed from an approximate function $q_\lambda(z|x) = N(z; \mu_\lambda, \sigma^2_\lambda)$, where $z = \mu_\lambda + \sigma_\lambda \odot \hat{\varepsilon}$ with $\hat{\varepsilon} \sim N(0, 1)$. Encoder E yields μ and σ, and with the re-parameterization trick, z is sampled from a Gaussian distribution. Then, with D', new samples are generated or real data samples are reconstructed; i.e., D' provides parameters for the real data distribution $p_\lambda(x|z)$. Subsequently, a sample drawn from $p_\phi(x|z)$ may be used to reconstruct the real data by marginalizing out z.

The GAN is an implicit generative model where a generator G and a discriminator D

compete in a minimax game over the training data in order to improve their performance. Generator G tries to approximate the underlying distribution of the training data and generates synthetic samples, while discriminator D learns to discriminate synthetic samples from real samples. The GAN model is trained on the following objectives:

$$\max_D V(D) = E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{x\sim p_g(z)}[\log(1 - D(G(z))]; \tag{3.2}$$

$$\min_G V(G) = E_{x\sim p_z(z)}[\log(1 - D(G(z))]. \tag{3.3}$$

G takes a noise sample $z \sim p_g(z)$ and learns to map it into image space as if it comes from the original data distribution $p_{\text{data}}(x)$, while D takes as input either real image data or generated image data and provides feedback to G as to whether that input is real or generated. On the one hand, D wants to maximize the likelihood for real samples and minimize the likelihood of generated samples; on the other hand, G wants D to maximize the likelihood of generated samples. A Nash equilibrium results when D can no longer distinguish real and generated samples, meaning that the model distribution matches the data distribution.

Makhzani et al. (2015) proposed the adversarial training of VAEs; i.e., VAE-GANs. Although they kept both D' and G, one can merge these networks since both can generate data samples from the noise samples of the representation z. In this case, D receives real data samples x and generated samples $\tilde{x}$ or $\hat{x}$ via G. Although G and D compete against each other, the feedback from D eventually becomes predictable for G and it keeps generating samples from the same class, at which point the generated samples lack heterogeneity. Figure 2.4c shows an example where all the generated images are of the same class. Durugkar et al. (2016) proposed that using multiple discriminators in a GAN model helps improve performance, especially for resolving this mode collapse. Moreover, a dynamic ensemble of multiple discriminators has recently been proposed to address the issue (Mordido et al., 2018).

As in a VAE-GAN, our MAVEN has three components, E, G, and D; all are CNNs with convolutional or transposed convolutional layers. First, E takes real samples x and

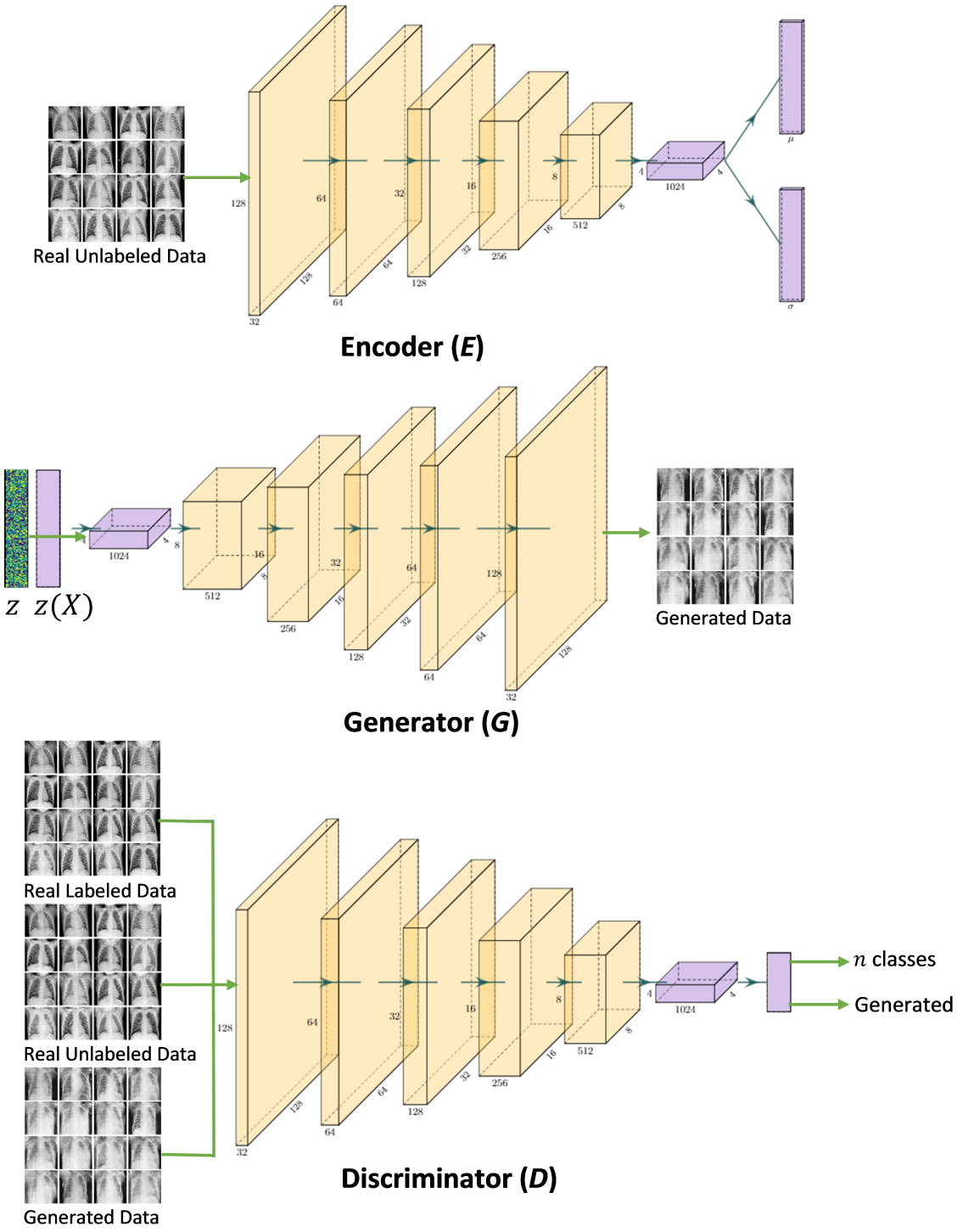


Figure 3.4: The three convolutional neural networks, E, G, and D, in the MAVEN.

generates a dimensionality-reduced representation $z(x)$. Second, G can input samples from noise distribution $z \sim p_g(z)$ or sampled noise $z(x) \sim q_\lambda(x)$ and it produces generated samples. Third, D takes inputs from distributions of real labeled data, real unlabeled data, and generated data. Fractionally strided convolutions are performed in G to obtain the image dimension from the latent code. The goal of an autoencoder is to maximize the Evidence Lower Bound (ELBO). The intuition here is to show the network more real data. The greater the quantity of real data that it sees, the more evidence is available to it and, as a result, the ELBO can be maximized faster.

In our MAVEN architecture (Figure 3.3), the VAE-GAN combination is extended to include multiple discriminators aggregated in an ensemble layer. K discriminators are collected and the combined feedback

$$V(D) = \frac{1}{K} \sum_{k=1}^{K} w_k D_k \tag{3.4}$$

is passed to G. In order to randomize the feedback from the multiple discriminators, a single discriminator is randomly selected.

3.2.1.2 Semi-Supervised Learning in MAVENs

Algorithm 3 presents the overall training procedure of our MAVEN model. In the forward pass, different real samples x into E and noise samples z into G provide different inputs for each of the multiple discriminators. In the backward pass, the combined feedback from the discriminators is computed and passed to G and E.

In the conventional image generator GAN, D works as a binary classifier—it classifies the input image as real or generated. To facilitate the training for an n-class classifier, D assumes the role of an $(n+1)$-classifier. For multiple logit generation, the sigmoid function is replaced by a softmax function. Now, it can receive an image x as input and output an $(n+1)$-dimensional vector of logits $\{l_1, \ldots, l_n, l_{n+1}\}$, which are finally transformed into class probabilities for the n labels in the real data while class $(n+1)$ denotes the

Algorithm 3: MAVEN Training procedure. m is the number of samples; B is the minibatch-size; and K is the number of discriminators.

steps $\leftarrow m/B$
for each **epoch do**
  for each step in *steps* **do**
    for k = 1 to **K do**
      Sample minibatch z_i; $z^{(1)}, \ldots, z^{(m)}$, $z_i \sim p_g(z)$
      Sample minibatch x_i; $x^{(1)}, \ldots, x^{(m)}$, $x_i \sim p_{\text{data}}(x)$
      Update discriminator D_k by ascending along its gradient:

$$\nabla_{\theta_{D_k}} \frac{1}{m} \sum_{i=1}^{m} [\log D_k(x_i) + \log(1 - D_k(G(z_i)))]$$

    end for
    Sample minibatch z_{k_i}; $1 \leq i \leq m$, $1 \leq k \leq K$, $z_{k_i} \sim p_g(z)$
    if ensemble is 'mean' **then**
      Assign weights w_k for each of the discriminators D_k
      Determine the mean discriminator D_μ of the discriminators $D_1, ..., D_k$

$$D_\mu = \frac{1}{K} \sum_{i}^{K} w_i D_i$$

    end if
    Update the generator G by descending along its gradient from the ensemble of discriminator D_μ:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D_\mu(G(z_i)))]$$

    Sample minibatch x_i; $x^{(1)}, \ldots, x^{(m)}$, $x_i \sim p_{\text{data}}(x)$
    Update encoder along its expectation function:

$$\nabla_{\theta_{E_{q_\lambda(z|x)}}} \left[\log \frac{p(z)}{q_\lambda(z \mid x)} \right]$$

  end for
end for

generated data. The probability that x is real and belongs to class $1 \leq i \leq n$ is

$$p(y = i \mid x) = \frac{\exp(l_i)}{\sum_{j=1}^{n+1} \exp(l_j)} \tag{3.5}$$

while the probability that x is generated corresponds to $i = n + 1$ in (3.5). As a semi-supervised classifier, the model takes labels only for a small portion of the training data. It is trained via supervised learning from the labeled data, while it learns in an unsupervised manner from the unlabeled data. The advantage comes from generating new samples. The model learns the classifier by generating samples from different classes.

3.2.1.3 Losses

Three networks, E, G, and D, are trained on different objectives. E is trained on maximizing the ELBO, G is trained on generating realistic samples, and D is trained to learn a classifier that classifies generated samples or particular classes for the real data samples.

D Loss: Since the model is trained on both labeled and unlabeled training data, the loss function of D includes both supervised and unsupervised losses. When the model receives real labeled data, it is the standard supervised learning loss

$$L_{D_{\text{supervised}}} = -\mathbb{E}_{x,y \sim p_{\text{data}}} \log[p(y = i \mid x)], \quad i < n + 1. \tag{3.6}$$

When it receives unlabeled data from three different sources, the unsupervised loss contains the original GAN loss for real and generated data from two different sources: syn$G$ directly from G and syn$E$ from E via G. The three losses,

$$L_{D_{\text{real}}} = -\mathbb{E}_{x \sim p_{\text{data}}} \log[1 - p(y = n + 1 \mid x)], \tag{3.7}$$

$$L_{D_{\text{syn}G}} = -\mathbb{E}_{\hat{x} \sim G} \log[p(y = n + 1 \mid \hat{x})], \tag{3.8}$$

$$L_{D_{\text{syn}E}} = -\mathbb{E}_{\tilde{x} \sim G} \log[p(y = n + 1 \mid \tilde{x})], \tag{3.9}$$

are combined as the unsupervised loss in D:

$$L_{D_{\text{unsupervised}}} = L_{D_{\text{real}}} + L_{D_{\text{syn}G}} + L_{D_{\text{syn}E}}. \tag{3.10}$$

G Loss: For G, the feature loss is used along with the original GAN loss. Activation $f(x)$ from an intermediate layer of D is used to match the feature between real and generated samples. Feature matching has shown much potential in semi-supervised learning (Salimans et al., 2016). The goal of feature matching is to encourage G to generate data that matches real data statistics. It is natural for D to find the most discriminative features in real data relative to data generated by the model:

$$L_{G_{\text{feature}}} = \|\mathbb{E}_{x\sim p_{\text{data}}} f(x) - \mathbb{E}_{\hat{x}\sim G} f(\hat{x})\|_2^2 . \tag{3.11}$$

The total G loss becomes the combined feature loss (3.11) plus the cost of maximizing the log-probability of D making a mistake on the generated data (syn$G$ / syn$E$); i.e.,

$$L_G = L_{G_{\text{feature}}} + L_{G_{\text{syn}G}} + L_{G_{\text{syn}E}}, \tag{3.12}$$

where

$$L_{G_{\text{syn}G}} = -\mathbb{E}_{\hat{x}\sim G} \log[1 - p(y = n+1 \mid \hat{x})], \tag{3.13}$$

and

$$L_{G_{\text{syn}E}} = -\mathbb{E}_{\tilde{x}\sim G} \log[1 - p(y = n+1 \mid \tilde{x})]. \tag{3.14}$$

E Loss: In the encoder E, the maximization of ELBO is equivalent to minimizing the KL-divergence, allowing approximate posterior inferences. Therefore the loss function includes the KL-divergence and also a feature loss to match the features in the syn$E$ data with the real data distribution. The loss for the encoder is

$$L_E = L_{E_{\text{KL}}} + L_{E_{\text{feature}}}, \tag{3.15}$$

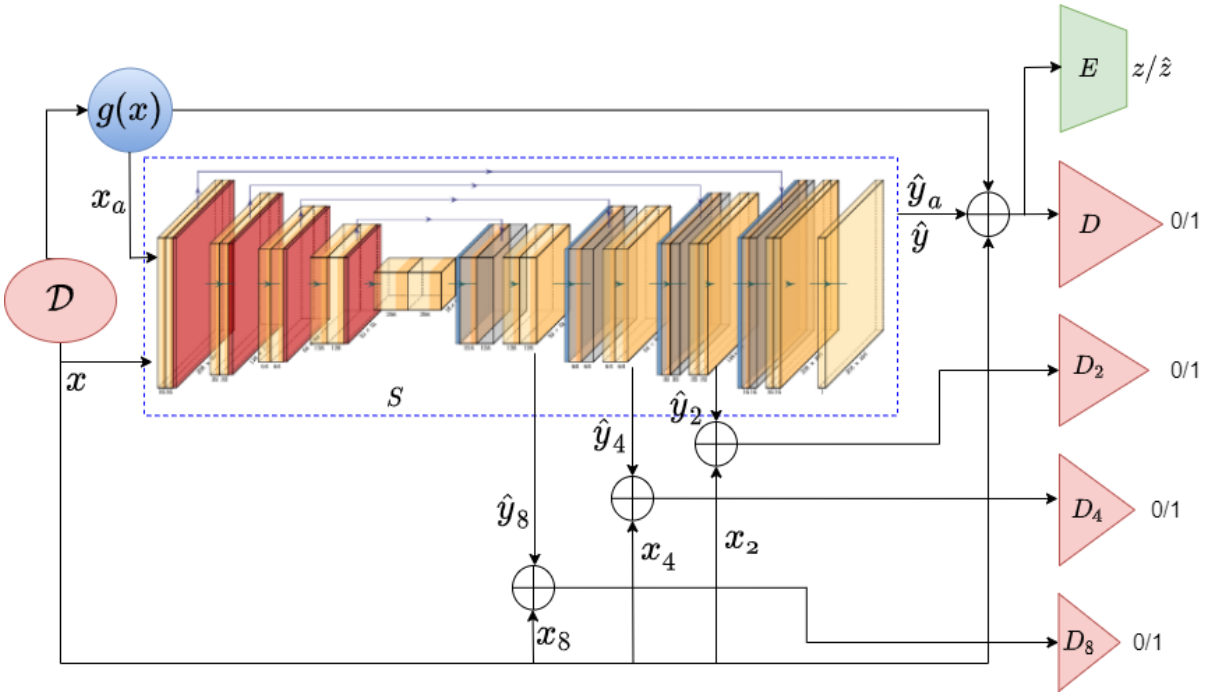


Figure 3.5: Schematic of the proposed PASS model: The segmentation mask generator S takes either input x or the transformed input x_a via a transformation function $g(x)$. The generated side outputs are passed to the corresponding discriminators D_2, D_4, and D_8 and the final outputs $y_a/\hat{y}$ are passed to the discriminator D. The shape encoder E also takes $y/\hat{y}$ as input to get the latent vector $z/\hat{z}$.

where

$$L_{E_{\mathrm{KL}}} = -\,\mathrm{KL}\left[q_\lambda(z \mid x) \,\|\, p(z)\right] = \mathbb{E}_{q_\lambda(z|x)}\left[\log \frac{p(z)}{q_\lambda(z \mid x)}\right] \approx \mathbb{E}_{q_\lambda(z|x)} \tag{3.16}$$

and

$$L_{E_{\text{feature}}} = \|\mathbb{E}_{x\sim p_{\text{data}}} f(x) - \mathbb{E}_{\tilde{x}\sim G} f(\tilde{x})\|_2^2 . \tag{3.17}$$

3.2.2 Domain Generalization Without Domain Specific Data

We next develop a novel model, namely Progressive Adversarial Semantic Segmentation (PASS), for improved segmentation across different data domains in medical imaging.

To formulate the problem, we assume an unknown data distribution $p(X, Y)$ over images and segmentation labels. The model has access to the labeled training set $\mathcal{D}$. and

unlabeled set $\mathcal{D}_{\mathcal{A}}$ through on-the-fly transformation from $p(X)$ after marginalizing out Y. We set the learning objectives for the segmentation task as:

$$\min_{\psi,\phi,\theta} \mathcal{L}_{\mathcal{L}}(\mathcal{D}, (\psi, \phi, \theta)) + \lambda \mathcal{L}_{\mathcal{A}}(\mathcal{D}_{\mathcal{A}}, (\psi, \phi, \theta)), \tag{3.18}$$

where the supervised objective $\mathcal{L}_{\mathcal{L}}$ is defined on the labeled data and the unsupervised objective $\mathcal{L}_{\mathcal{A}}$ is defined on the unlabeled data, λ is a non-negative weight parameter, and ψ, ϕ, and θ denote the parameters of the segmentor S, discriminator D, and encoder E networks, respectively.

3.2.2.1 Progressive Adversarial Semantic Segmentation (PASS)

Figure 3.5 illustrates the PASS model. The training algorithm is detailed in Algorithm 4. PASS is based on the backbone of a progressive U-Net with some careful adjustments in the U-Net with side-adversary and side-supervision capabilities. As in a U-Net (Ronneberger et al., 2015b), PASS has a segmentor (S) with skip connections in an encoder-decoder architecture.

In each encoder layer, two 3×3 convolutions are followed by instance normalization, leaky-ReLU activation, and a 2×2 max-pooling. We generate side-outputs in every stage of the decoder. The side-outputs are collected at the resolutions of $x/8$, $x/4$, and $x/2$ before the final output at the resolutions of x. According to the shape of the side-outputs, discriminators are employed and layers are added progressively. Progressively adding one side-output to the next improves the segmentation performance compared to collecting the output from the final decoder stage (Imran et al., 2018). The progressive side-outputs also ensure that the network does not lose track of objects of interest. Moreover, progressively growing the discriminators enables the model to receive feedback at different resolutions via the side-outputs. Tables 3.3-3.6 detail the architectures of the discriminators D, $D2$, $D4$, and $D8$. While the discriminators are employed at different side-outputs, the segmentor tries to generate side-outputs closer to the ground truths progressively for an improved and accurate final segmentation. A shape encoder E (detail architecture in Table 3.2)

Algorithm 4: PASS Training.

Require:

Training set of labeled data $x, y \in \mathcal{D}$
Transformation function $g(x)$ to generate x_a from x
Network architecture $S_\phi, D_\psi.E_\theta \in \mathcal{F}_{(\phi,\psi,\theta)}$
with learnable parameters ϕ, ψ, θ

for each **epoch** over $\mathcal{D}$ **do**

Generate minibatches of unlabeled inputs $\mathcal{M}_\mathcal{A}$ using $g(x)$:
$x_a = g(x)$

for each **step do**

Sample minibatch $\mathcal{M}$: $x_{(i)}; x_{(1)}, \ldots, x_{(m)} \sim p_{\mathcal{D}(x)}$

Compute model outputs for the labeled inputs:
$\hat{y} \leftarrow \mathcal{F}_(\phi, \psi, \theta(\mathcal{M})$
Compute model outputs for the unlabeled inputs:
$\hat{y}_a \leftarrow \mathcal{F}_\phi, \psi, \theta(\mathcal{M}_\mathcal{A})$

Update the discriminators D_i, $i = 1, \ldots, d$, along their gradients:

$$\nabla_{\psi_{D_i}} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \left[L_{D_i\left(x_{(i)}, y_{(i)}, \hat{y}_{(i)}\right)} \right] + \alpha \frac{1}{|\mathcal{M}_\mathcal{A}|} \sum_{i \in \mathcal{M}_\mathcal{A}} \left[L_{D_i\left(x_{a_{(i)}}, \hat{y}_{a_{(i)}}\right)} \right].$$

Update the shape encoder E along its gradient:

$$\nabla_{\theta_E} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \left[L_{E_{\left(x_{(i)}, y_{(i)}, \hat{y}_{(i)}\right)}} \right].$$

Update the segmentation mask generator S along its gradient:

$$\nabla_{\phi_S} \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \left[L_{S_{\left(y_{(i)}, \hat{y}_{(i)}\right)}} \right] + \alpha \frac{1}{|\mathcal{M}_\mathcal{A}|} \sum_{i \in \mathcal{M}_\mathcal{A}} \left[L_{S_{\left(y_{(i)}, \hat{y}_{a_{(i)}}\right)}} \right].$$

end for
end for

Table 3.2: Architecture details of the shape encoder (E)

Name	Feature maps (input)	Feature maps (output)
Conv layer - 1a	$256 \times 256 \times 3$	$256 \times 256 \times 16$
Conv layer - 1b	$256 \times 256 \times 16$	$256 \times 256 \times 16$
Max pool - 1	$256 \times 256 \times 16$	$128 \times 128 \times 16$
Conv layer - 2a	$128 \times 128 \times 16$	$128 \times 128 \times 32$
Conv layer - 2b	$128 \times 128 \times 32$	$128 \times 128 \times 32$
Max pool - 2	$128 \times 128 \times 32$	$64 \times 64 \times 32$
Conv layer - 3a	$64 \times 64 \times 32$	$64 \times 64 \times 64$
Conv layer - 3b	$64 \times 64 \times 64$	$64 \times 64 \times 64$
Max pool - 3	$64 \times 64 \times 64$	$32 \times 32 \times 64$
Conv layer - 4a	$32 \times 32 \times 64$	$32 \times 32 \times 128$
Conv layer - 4b	$32 \times 32 \times 128$	$32 \times 32 \times 128$
Max pool - 4	$32 \times 32 \times 128$	$16 \times 16 \times 128$
Conv layer - 5a	$16 \times 16 \times 128$	$16 \times 16 \times 256$
Conv layer - 5b	$16 \times 16 \times 256$	$16 \times 16 \times 256$
encoder flatten - 5	$16 \times 16 \times 256$	65,536
encoder dense - z	65,536	256

is used in PASS to match the latent representation of the stacked input and output of S with the stacked input and reference so that the model becomes shape-aware while mapping an input to the segmentation mask. Moreover, during training, a transformation function is used to obtain x_a from input x, and PASS makes a segmentation prediction on x_a. Note that x_a is used without any corresponding label information. Through this, PASS is trained on an enlarged scope of the data distribution, useful for cross-domain predictions.

3.2.2.2 Loss Functions

The three networks S, D, and E in the PASS model, are trained on separate objectives:

Segmentor Loss: The objective of segmentor S is based on the segmentation maps generated in different resolutions. We chose Dice loss to penalize the model for the segmentation map predictions. Therefore, the objective of S includes segmentation loss as a weighted sum of all the side-output and side-adversarial losses, where S wants the discriminators D_i to maximize the likelihood for the predicted segmentations. For

Table 3.3: Architecture details of the Discriminator (D)

Name	Feature maps (input)	Feature maps (output)
Conv layer - 1a	$256 \times 256 \times 3$	$256 \times 256 \times 16$
Conv layer - 1b	$256 \times 256 \times 16$	$256 \times 256 \times 16$
Max pool - 1	$256 \times 256 \times 16$	$128 \times 128 \times 16$
Conv layer - 2a	$128 \times 128 \times 16$	$128 \times 128 \times 32$
Conv layer - 2b	$128 \times 128 \times 32$	$128 \times 128 \times 32$
Max pool - 2	$128 \times 128 \times 32$	$64 \times 64 \times 32$
Conv layer - 3a	$64 \times 64 \times 32$	$64 \times 64 \times 64$
Conv layer - 3b	$64 \times 64 \times 64$	$64 \times 64 \times 64$
Max pool - 3	$64 \times 64 \times 64$	$32 \times 32 \times 64$
Conv layer - 4a	$32 \times 32 \times 64$	$32 \times 32 \times 128$
Conv layer - 4b	$32 \times 32 \times 128$	$32 \times 32 \times 128$
Max pool - 4	$32 \times 32 \times 128$	$16 \times 16 \times 128$
discriminator flatten - 4	$16 \times 16 \times 128$	32768
discriminator dense - l	32768	1

Table 3.4: Architecture details of the Discriminator (D_2)

Name	Feature maps (input)	Feature maps (output)
Conv layer - 1a	$128 \times 128 \times 3$	$128 \times 128 \times 32$
Conv layer - 1b	$128 \times 128 \times 32$	$128 \times 128 \times 32$
Max pool - 1	$128 \times 128 \times 32$	$64 \times 64 \times 32$
Conv layer - 2a	$64 \times 64 \times 32$	$64 \times 64 \times 64$
Conv layer - 2b	$64 \times 64 \times 64$	$64 \times 64 \times 64$
Max pool - 2	$64 \times 64 \times 64$	$32 \times 32 \times 64$
Conv layer - 3a	$32 \times 32 \times 64$	$32 \times 32 \times 128$
Conv layer - 3b	$32 \times 32 \times 128$	$32 \times 32 \times 128$
Max pool - 3	$32 \times 32 \times 128$	$16 \times 16 \times 128$
discriminator flatten - 3	$16 \times 16 \times 128$	32768
discriminator dense - l	32768	1

Table 3.5: Architecture details of the Discriminator (D_4)

Name	Feature maps (input)	Feature maps (output)
Conv layer - 1a	$64 \times 64 \times 3$	$64 \times 64 \times 64$
Conv layer - 1b	$64 \times 64 \times 64$	$64 \times 64 \times 64$
Max pool - 1	$64 \times 64 \times 64$	$32 \times 32 \times 64$
Conv layer - 2a	$32 \times 32 \times 64$	$32 \times 32 \times 128$
Conv layer - 2b	$32 \times 32 \times 128$	$32 \times 32 \times 128$
Max pool - 2	$32 \times 32 \times 128$	$16 \times 16 \times 128$
discriminator flatten - 2	$16 \times 16 \times 128$	32768
discriminator dense - l	32768	1

Table 3.6: Architecture details of the Discriminator (D_8)

Name	Feature maps (input)	Feature maps (output)
Conv layer - 1a	$32 \times 32 \times 3$	$32 \times 32 \times 128$
Conv layer - 1b	$32 \times 32 \times 128$	$32 \times 32 \times 128$
Max pool - 1	$32 \times 32 \times 128$	$16 \times 16 \times 128$
discriminator flatten - 1	$16 \times 16 \times 128$	32768
discriminator dense - l	32768	1

the segmentation predictions, we employ Dice losses and the final loss is calculated as $\mathcal{L}_{S_{seg}} = \sum_i^4 w_i \mathcal{L}_{(y_i, \hat{y}_i)}$. A second segmentation loss term is used for logit-wise distribution comparison. Since x_a is not paired with any reference segmentations, it is not possible to directly compare segmentation loss. Rather, we employ Kullback-Leibler (KL) divergence to penalize S for not maintaining the distribution of the predicted segmentation of the labeled data. The KL loss is calculated as

$$\mathcal{L}_{S_{\mathrm{KL}}} = \sum_i^{m^2} \left| (\hat{y}_{pk}(i) - \hat{y}_{a_{pk}}(i)) \log(y_{pk}(i)/\hat{y}_{a_{pk}}(i)) \right|. \tag{3.19}$$

Then, the segmentor's adversarial loss is calculated from the stacked input image and predicted segmentation when S wants D to maximize the likelihood as

$$\mathcal{L}_{S_{pred_{(x_i, \hat{y}_i)}}} = -\mathbb{E}_{x_i, \hat{y}_i \sim S} \log[1 - D_i(x_i, \hat{y}_i)]. \tag{3.20}$$

Since the main objective of the segmentor is to generate the segmentation map, $L_{S_{pred}}$

is usually weighed using a small number α. In addition, a feature loss is calculated by collecting intermediate convolutions from the discriminators. The goal of feature matching is to push S to generate segmentations that match reference data statistics. It is natural that D can find the most discriminative features. The feature loss across all the discriminators is calculated and summed as

$$\mathcal{L}_{S_{feature}} = \sum_{i}^{d} ||f_{x,y\sim\mathcal{D}}(x_i, y_i) - f_{x,\hat{y}\sim S}(x_i, \hat{y}_i)||_2^2. \tag{3.21}$$

Discriminator Loss: The discriminator has only unsupervised loss objectives. When the model receives the stacked input image and reference segmentation label (x, y) from two different sources, the unsupervised loss contains the original adversarial loss for real data:

$$\mathcal{L}_{D_{i_{real}}} = -\mathbb{E}_{x_i,y_i\sim p_{data}} \log[1 - D_i(x_i, y_i)]. \tag{3.22}$$

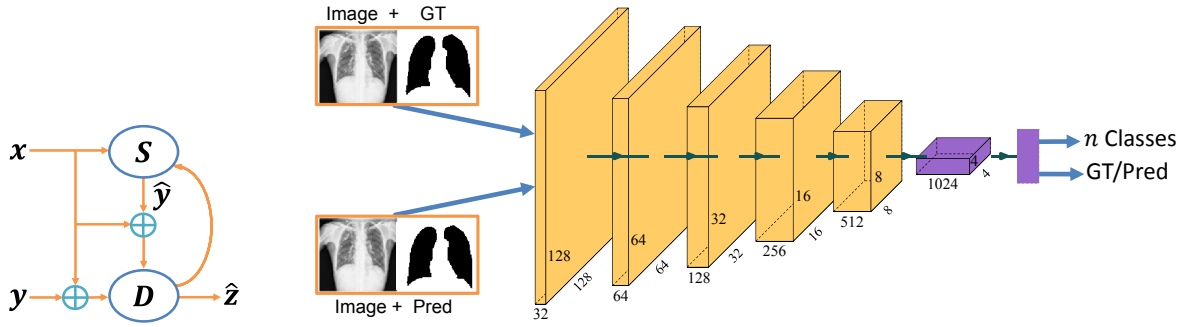
Similarly, the adversarial loss for predicted data is calculated from the stacked input image and predicted segmentation label $x, \hat{y}$ as follows:

$$\mathcal{L}_{D_{i_{pred}}} = -\mathbb{E}_{(x_i,\hat{y}_i)\sim S} \log[D_i(x_i, \hat{y}_i)]. \tag{3.23}$$

Encoder Loss: The encoder is trained on matching the shapes of the predicted masks with the reference masks. The encoder E is fed with input image stacked with either the reference mask or the predicted mask, and their latent representations are acquired as the outputs. The encoder loss is simply the mean-square error between the two latent representations:

$$\mathcal{L}_E = \frac{1}{n} \sum_{i}^{n} ||z - \hat{z}||, \tag{3.24}$$

where z is the latent vector representation of the reference segmentation and $\hat{z}$ is the latent vector representation of the predicted segmentation.



(a) (b)

Figure 3.6: (a) Basic structure of the proposed APPAU-Net model. The segmentor S predicts segmentation $\hat{y}$ from a given image x. The discriminator D predicts the class label $\hat{z}$ from image-real label pair (x, y); $z = 0, \ldots, n$ are real disease classes and $z = n+1$ denotes the predicted class; (b) Detailed architecture of the Discriminator D (as a CNN) network of the APPAU-Net model.

3.2.3 Semi-Supervised Multitask Learning

3.2.3.1 Adversarial Pyramid Progressive Attention U-Net (APPAU-Net)

Our proposed APPAU-Net model consists of two major building blocks, a segmentor S and a discriminator D (Figure 3.6). S consists of a pyramid encoder and a progressive attention-gated decoder modifying a U-Net. The S network, which is illustrated in Figure 3.7, receives the image input x at different scales in different stages of the encoder (Fu et al., 2018). This pyramidal input allows the model to access class details at different scales. Moreover, while lowering resolution, the model can keep track of the ROIs, avoiding the possibility of losing them after the subsequent convolutions. The pyramid input to the encoder network enables the model to learn more locally-aware features crucial to semantic segmentation.

Following Imran et al. (2018), with deep-supervision, APPAU-Net generates side-outputs at different resolutions from the decoder. The side-outputs are progressively added to the next side-outputs before reaching the final segmentation at the original image resolution. Combining pyramid inputs and progressive side-outputs helps the model perform better in segmenting small ROIs. The side-output segmentation maps $\hat{y}_i$ are compared to the ground truth mask to calculate the side-losses of varying weights (higher

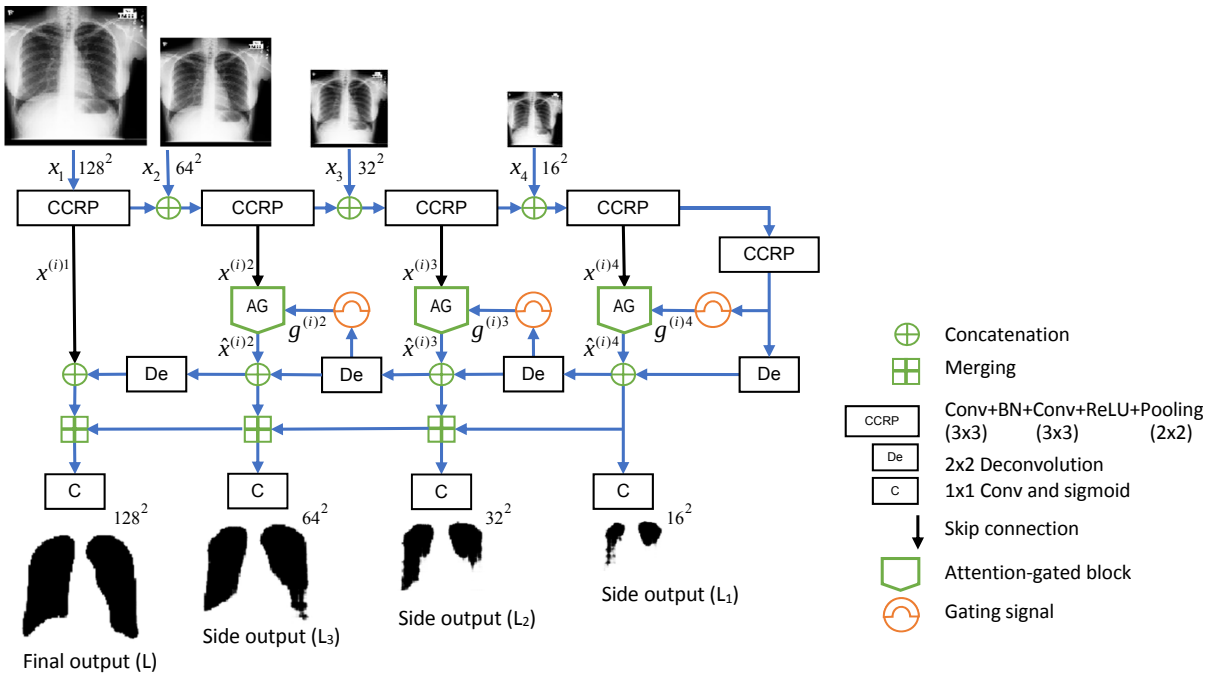


Figure 3.7: Architecture of the segmentor or PPAU-Net in our APPAU-Net model. The encoder takes inputs at different scales and progressively adds the side-outputs from the attention-gated decoder. The discriminator takes image-label or image-predicted label pairs and classifies the images.

resolutions are usually assigned higher weights). Therefore, the final segmentation loss is calculated as

$$L_{seg_{(x,y)}} = \sum_{i=1}^{4} w_i L_{(y_i,\hat{y}_i)}, \tag{3.25}$$

where, w_i's corresponds to the weights assigned to the losses L_i's.

However, generating segmentation maps (side-outputs) at different stages of the decoder might lead to loss of spatial detail. In cases with substantial shape variability of the ROIs, this eventually incurs larger false positives. To tackle this problem, we adapt soft-attention gates that help draw relevant spatial features from the low-level feature maps of the pyramid encoder (Oktay et al., 2018). Feature maps are then propagated to the high-level features to generate side-outputs at different stages of the decoder. Attention-gated (AG) modules produce attention coefficients $\alpha \in [0, 1]$ at each pixel i that scale input feature maps $x^{(i)l}$ at layer l to semantically relevant features $\hat{x}^{(i)l}$. A gating signal from coarser resolution, serves to determine the focus regions through the

computation of intermediate maps, as follows:

$$G^l_{\text{attn}} = \psi^T(\sigma(w_x^T x^{(i)l} + w_g^T g^{(i)l} + b_g)) + b_\psi. \tag{3.26}$$

The linear attention coefficients are computed by element-wise summation and a 1×1 linear transformation. The parameters are w_x, w_g, b_g, and b_ψ. The intermediate maps are then transformed using ReLU σ_1 and sigmoid σ_2 activations. Finally, after element-wise multiplication of the feature map $x^{(i)l}$ (via skip) and nonlinear transformation, $\hat{x}^{(i)l}$ is generated at each decoder stage.

The attention coefficients α_i retain the relevant features by scaling the low level query signal $x^{(i)l}$ through an element-wise product. These pruned features are then concatenated with upsampled output maps at different stages of the decoder. A 1×1 convolution and sigmoid activation is applied on each output map in the decoder to generate the side-outputs at different resolutions. With deep supervision and gating from the pyramid encoder, the model becomes semantically more discriminative.

3.2.3.2 Loss Functions

The overall training procedure of the proposed APPAU-Net model is presented in Algorithm 5. The real samples and labels to S are presented in the forward pass. In the backward pass, the feedback from D is determined and passed to S. Similar to the semi-supervised learning of MAVENs(3.2.1.2), D in APPAU-Net is trained as $(n+1)$-class classifier. The two building blocks of our APPAU-Net model have different objectives.

Segmentor Loss: As in the semi-supervised learning-scheme, the segmentor's objective is just based on the labeled samples. We employ Tversky loss, a generalization of Dice loss that weighs false negatives higher than than false positives in order to balance precision and recall. The segmentor's objective includes a segmentation loss and an adversarial loss, where the segmentor wants the discriminator D to maximize the likelihood for the predicted segmentation generated by the segmentor. We combine an absolute KL

Algorithm 5: Adversarial Pyramid Progressive Attention U-Net (APPAU-Net) Training Procedure. m is the number of samples and b is the minibatch-size.

$steps \leftarrow \frac{m}{b}$
for each **epoch do**
  for each step in *steps* **do**
    Sample minibatch $y_i; y^{(1)}, \ldots, y^{(m)}, y_i \sim p_{\text{data}}(y)$
    Sample minibatch $x_i; x^{(1)}, \ldots, x^{(m)}, x_i \sim p_{\text{data}}(x)$
    Update discriminator D by ascending along its gradient:

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i, y_i) + \log(1 - D(x_i, S(x_i))]$$

    Sample minibatch $x_i; x^{(1)}, \ldots, x^{(m)}, x_i \sim p_{\text{data}}(x)$
    Update the segmentor S by descending along its gradient from the discriminator D and also the segmentation loss (depending on the choice of loss function):

$$\nabla_{\theta_S} \frac{1}{m} \sum_{i=1}^{m} \Big[\log(1 - D(x_i, S(x_i))) + L_{seg_{(i)}} \Big]$$

  end for
end for

divergence with a Tversky loss, proposing the new loss function

$$L_S = L_{S_{seg_{(y,\hat{y})}}} + cL_{S_{adv_{(x,\hat{y})}}}, \tag{3.27}$$

where $L_{S_{seg_{(y,\hat{y})}}} = aL_{S_{KL}} + bL_{S_{TV}}$, with $L_{S_{KL}} = \sum_i^{m^2} |(y_{pl}(i) - \hat{y}_{pl}^{(i)}) \log(y_{pl}^{(i)} / \hat{y}_{pl}^{(i)})|$, and

$$L_{S_{TV}} = 1 - \frac{\sum_i^{m^2} y_{pl}^{(i)} \hat{y}_{pl}^{(i)} + \epsilon}{\sum_i^{m^2} y_{pl}^{(i)} \hat{y}_{pl}^{(i)} + \alpha \sum_i^{m^2} y_{p\bar{l}}^{(i)} \hat{y}_{pl}^{(i)} + \beta \sum_i^{m^2} y_{pl}^{(i)} \hat{y}_{p\bar{l}^{(i)}} + \epsilon}, \tag{3.28}$$

where $\hat{y}_{pl}(i)$ is the prediction probability that pixel i is assigned label l (one of the ROI labels) and $\hat{y}_{p\bar{l}}(i)$ is the probability that the pixel i is assigned the non-ROI (background) label. Similarly, $y_{pl}(i)$ and $y_{p\bar{l}}(i)$ denote the pixel-wise mapping labels in the ground-truth masks. Hyper parameters a, b, α, and β can be tuned to weigh the KL-divergence against the Tversky loss (first pair) and weigh FPs against FNs. Small constant ϵ avoids division by zero. The second term in the segmentor's objective is an adversarial loss, where the

segmentor wants the discriminator to maximize likelihood for the paired data x and predicted segmentation $\hat{y}$. Therefore, the segmentor's adversarial loss is

$$L_{S_{adv_{(x,\hat{y})}}} = -\mathbb{E}_{x,\hat{y}\sim S} \log[1 - p(z = n + 1 \mid (x, \hat{y})]. \tag{3.29}$$

Since the main objective of the segmentor is to generate the segmentation map, $L_{S_{adv}}$ is usually weighed using a small number c.

Discriminator Loss: The discriminator is trained on multiple objectives—adversary on the segmentor's output and classification of the images into one of the real classes. Since the model is trained on both labeled and unlabeled training data, the loss function of the discriminator D includes both supervised and unsupervised losses. When the model receives image-label pairs (x, y), it is just the standard supervised learning loss

$$L_{D_{sup}} = -\mathbb{E}_{x,y,z\sim p_{data}} \log[p(z = i \mid x, y; i < n + 1)]. \tag{3.30}$$

When it receives unlabeled data (x, y) or $(x, \hat{y})$ from two different sources, the unsupervised loss combines the original adversarial losses for image-real label and image-prediction pairs:

$$L_{D_{label}} = -\mathbb{E}_{x,y\sim p_{data}} \log[1 - p(z = n + 1 \mid x, y)] \tag{3.31}$$

and

$$L_{D_{pred}} = -\mathbb{E}_{(x,\hat{y})\sim S} \log[p(z = n + 1 \mid x, \hat{y})]. \tag{3.32}$$

3.2.4 Self-Supervised, Semi-Supervised, Multi-Task Learning

We next combine the concepts of self-supervision and adversarial learning in a semi-supervised multitask learning scheme for jointly performing image-level prediction or pixel-wise classifications within the same model. To formulate the problem, we assume an unknown data distribution $p(X, Y, C)$ over images, segmentation labels, and class labels. The model has access to the labeled training set $\mathcal{D}_{\mathcal{L}}$ sampled i.i.d. from $p(X, Y, C)$ and to

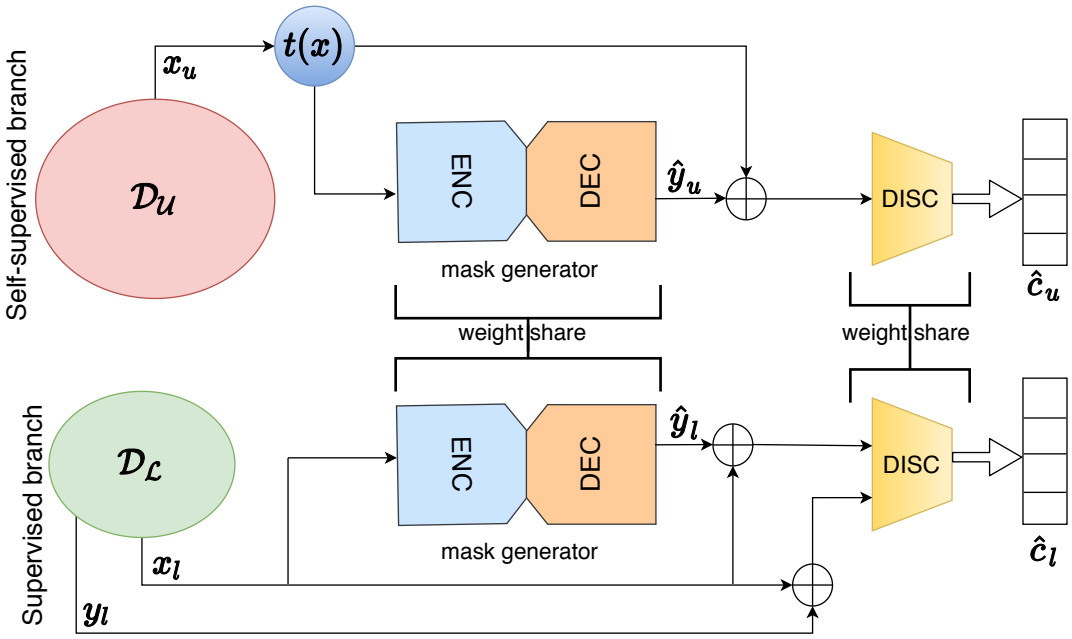


Figure 3.8: Schematic of the self-supervised, semi-supervised, multitask learning model (S^{4}MTL). A segmentation mask generator produces masks on taking inputs from labeled or unlabeled samples. A class discriminator takes concatenated inputs from labeled data-mask or unlabeled data-mask pairs and predicts the class labels. For the labeled data branch, it is fully supervised. By contrast, using unlabeled data, the self-supervised branch employs self-generated labels using a geometric transformation function $t(x)$. The predicted segmentation output is obtained from the decoder (DEC) and the diagnostic classification prediction is received at the discriminator (DISC).

the unlabeled training set $\mathcal{D}_\mathcal{U}$ sampled i.i.d. from $p(X)$ after marginalizing out Y and C. We set the learning objectives for both classification and segmentation tasks as

$$\min_{\psi,\theta} \; \mathcal{L}_\mathcal{L}(\mathcal{D}_\mathcal{L}, (\psi, \theta)) + \alpha \mathcal{L}_\mathcal{U}(\mathcal{D}_\mathcal{U}, (\psi, \theta)), \tag{3.33}$$

where the supervised loss $\mathcal{L}_\mathcal{L}$ is defined on the labeled data and the unsupervised loss $\mathcal{L}_\mathcal{U}$ is defined on the unlabeled data, α is a non-negative weight parameter, and ψ and θ denote the parameters of the classification and the segmentation networks, respectively.

Referring to Figure 3.8, the S^{4}MTL model comprises two main components—a segmentation mask generator G and a class discriminator D. Generator G can be any segmentation network, such as U-Net (Ronneberger et al., 2015b), and any CNN classifier (LeCun et al., 2010) may be used as discriminator D. We employed an encoder-decoder

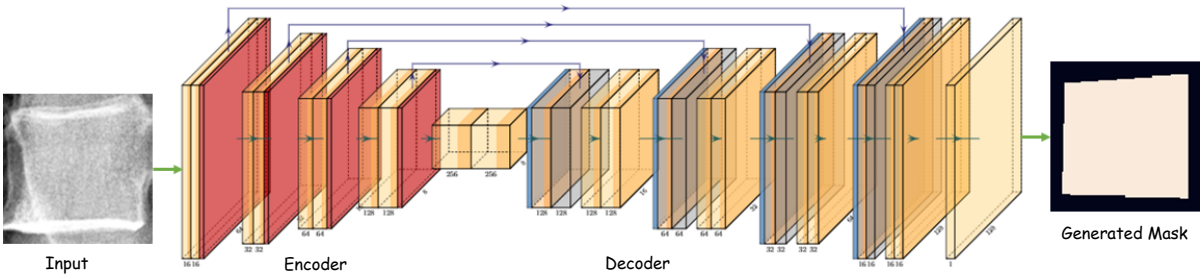


Figure 3.9: Detailed architecture of the segmentation mask generator (G) network.

architecture with skip connections as shown in Figure 3.9 for the segmentation mask generator network G and the previous architecture (Figure 3.6b) for the class discriminator network D. The model has a supervised branch for the labeled data and a self-supervision branch for the unlabeled data, and the two branches share the same G and D. Labeled data are passed through the supervised branch and supervised losses are calculated. Unsupervised losses are computed by feeding the unlabeled data through the self-supervised branch. The two networks G and D are trained in an adversarial learning manner, where the mask generator and the class discriminator compete against each other. The objective in (3.33) is therefore specified as two losses $\mathcal{L}_D$ and $\mathcal{L}_G$ for the two networks D and G, respectively:

$$\begin{aligned} &\min_{\psi} \ \mathcal{L}_D(\mathcal{L}(\mathcal{D}_{\mathcal{L}}, \psi_D) + \alpha\mathcal{L}(\mathcal{D}_{\mathcal{U}}, \psi_D)), \\ &\min_{\theta} \ \mathcal{L}_G(\mathcal{L}(\mathcal{D}_{\mathcal{L}}, \theta_G) + \alpha\mathcal{L}(\mathcal{D}_{\mathcal{U}}, \theta_G)). \end{aligned} \tag{3.34}$$

Algorithm 6 specifies the overall training procedure of the S^{4}MTL model.

3.2.4.1 Self-Supervision

Self-supervision is usually formulated on two tasks—a surrogate or pretext task and a downstream or main task. Unlike fine-tuning on the downstream task using the pre-trained model from the pretext task, we combine them and refer to the combination as the *pre-stream* (i.e., *Pre*text + down*stream*) task for concurrently performing supervision and self-supervision. Although the self-supervision can be applied to both labeled and unlabeled data, we confine self-supervision only to the unlabeled data. We define pretext

Algorithm 6: S^{4}MTL Mini-Batch Training.

Require:

Training set of labeled data $x_l, y_l, c_l \in \mathcal{D}_\mathcal{L}$

Training set of unlabeled inputs $x_u \in \mathcal{D}_\mathcal{U}$

Transformation function $t(x)$ to generate c_u from x_u

Network architecture $D_\psi, G_\theta \in \mathcal{F}_{(\psi,\theta)}$ with learnable parameters ψ, θ

for each epoch over $\mathcal{D}_\mathcal{U}$ **do**

Generate minibatches of unlabeled inputs $\mathcal{M}_\mathcal{U}$ using $t(x)$

for each step **do**

Sample minibatch $x_{l_{(i)}}; x_{l_{(1)}}, \ldots, x_{l_{(m)}} \sim p_{\mathcal{D}_\mathcal{L}}(x)$

Sample minibatch $x_{u_{(i)}}; x_{u_{(1)}}, \ldots, x_{u_{(m)}} \sim p_{\mathcal{D}_\mathcal{U}}(x)$

Compute model outputs for the labeled inputs:

$\hat{y}_l, \hat{c}_l \leftarrow \mathcal{F}_{(\psi,\theta)}(\mathcal{M}_\mathcal{L})$

Compute model outputs for the unlabeled inputs:

$\hat{y}_u, \hat{c}_u \leftarrow \mathcal{F}_{(\psi,\theta)}(\mathcal{M}_\mathcal{U})$

Update the class discriminator D along its gradient:

$$\nabla_{\psi_D} \frac{1}{|\mathcal{M}_\mathcal{L}|} \sum_{i \in \mathcal{M}_\mathcal{L}} \left[L_{D_{\left(x_{l_{(i)}}, y_{l_{(i)}}, \hat{y}_{l_{(i)}}, c_{l_{(i)}}, \hat{c}_{l_{(i)}}\right)}} \right] + \alpha \frac{1}{|\mathcal{M}_\mathcal{U}|} \sum_{i \in \mathcal{M}_\mathcal{U}} \left[L_{D_{\left(x_{u_{(i)}}, \hat{y}_{u_{(i)}}, \hat{c}_{u_{(i)}}\right)}} \right]$$

Update the segmentation mask generator G along its gradient:

$$\nabla_{\theta_G} \frac{1}{|\mathcal{M}_\mathcal{L}|} \sum_{i \in \mathcal{M}_\mathcal{L}} \left[L_{S_{\left(x_{l_{(i)}}, y_{l_{(i)}}, \hat{y}_{l_{(i)}}\right)}} \right] + \alpha \frac{1}{|\mathcal{M}_\mathcal{U}|} \sum_{i \in \mathcal{M}_\mathcal{U}} \left[L_{G_{\left(x_{u_{(i)}}, \hat{y}_{u_{(i)}}\right)}} \right]$$

end for

end for

tasks for the unlabeled data $\mathcal{D}_\mathcal{U}$. The self-supervision applies to the class discriminator D, whereas it is still unsupervised at the mask generator G. For the classification, we use a transformation function $t(x)$ to randomly flip (horizontal/vertical) or rotate (0, 90, 180, etc.) the unlabeled images and allow the network D predict them.

3.2.4.2 Classification

The real samples and labels to G are presented in the forward pass. In the backward pass, the feedback from D is passed to G. In order to facilitate the training of an n-class classifier, the class discriminator D is trained to perform as an $(n+1)$-classifier. For multiple logit

generation, we utilize softmax function, such that it can receive concatenated image-mask (x_l, y_l), image-predicted mask $(x_l, \hat{y_l})$ for labeled data, and $(x_u, \hat{y_u})$ for unlabeled data as inputs, and output an $(n+1)$-dimensional vector of logits $\{l_1, l_2, \ldots, l_{n+1}\}$, which are finally transformed into class probabilities for the final classification. The class probabilities for the labeled data are calculated as

$$p(\hat{c_l} = (c_l = i) \mid (x_l, y_l)) = \frac{\exp(l_i)}{\sum_{j=1}^{n+1} \exp(l_j)} \tag{3.35}$$

and for the unlabeled data as

$$p(\hat{c_u} = (c_u = i) \mid (x_u, \hat{y}_u)) = \frac{\exp(l_i)}{\sum_{j=1}^{n+1} \exp(l_j)}. \tag{3.36}$$

3.2.4.3 Segmentation

The segmentation mask generator takes input x_l and generates $\hat{y}_l$ for the labeled data. For the unlabeled data, mask prediction $\hat{y}_u$ is generated from input x_u. For the labeled data, it is just like regular supervised segmentation. We used Dice loss for the base model. Since the ground truth mask y_l is also available, the segmentation loss is calculated as

$$L_{G_{\mathcal{L}}(\text{seg})} = 1 - \frac{\sum_i^{m^2} y_{pk}^{(i)} \hat{y}_{pk}^{(i)}}{\sum_i^{m^2} y_{pk}^{(i)} \hat{y}_{pk}^{(i)} + \frac{1}{2} \sum_i^{m^2} y_{p\bar{k}}^{(i)} \hat{y}_{pk}^{(i)} + \frac{1}{2} \sum_i^{m^2} y_{pk}^{(i)} \hat{y}_{p\bar{k}^{(i)}}}. \tag{3.37}$$

On the other hand, the ground truth mask for the unlabeled data is not available; therefore, we cannot directly calculate the segmentation loss on the predicted mask $\hat{y}_u$. Instead, we used two unsupervised losses—unpaired consistency regularization loss and logit-wise KL divergence. The logit-wise absolute KL divergence (Imran and Terzopoulos, 2019b) between the ground truth of labeled data y_l and prediction on unlabeled data $\hat{y}_u$ over each pixel i and logit k is

$$L_{G_{\text{KL}}(\mathcal{D}_U)} = \sum_i^{m^2} \left| (y_{l_{pk}}(i) - \hat{y}_{u_{pk}}(i)) \log(y_{l_{pk}}(i) / \hat{y}_{u_{pk}}(i)) \right|. \tag{3.38}$$

The consistency loss for the prediction $\hat{y}_u$ of the unlabeled data against prediction $\hat{y}_l$ of the labeled data is

$$L_{G_{\text{cyc}}(\mathcal{D}_U)} = \mathbb{E}_{\hat{y}_u} \sim p(\mathcal{D}_\mathcal{L})[||\hat{y}_l - \hat{y}_u||_1] + \mathbb{E}_{\hat{y}_l} \sim p(\mathcal{D}_\mathcal{U})[||\hat{y}_u - \hat{y}_l||_1]. \tag{3.39}$$

3.2.4.4 Final Losses

Depending on the source of the model inputs, G and D have multiple objectives for labeled and unlabeled data, combined for training with gradient descent.

Like any supervised learning model, G's supervised loss is just based on the labeled samples (at pixel-level). We employ the generalized Dice loss in this regard. As in adversarial training, the generator's objective includes segmentation loss and adversarial prediction loss, where the segmentation mask generator G wants the class discriminator D to maximize the likelihood for the generated segmentation masks. For the labeled examples, we calculate two-way losses from image-label and image-prediction pairs, which differs from the unlabeled examples, where only image-prediction pairs are taken into account. The segmentation loss terms are calculated using Eqns. (3.37)–(3.39). The unsupervised adversarial prediction loss terms include adversarial prediction losses for the labeled and unlabeled data. The mask generator G wants the class discriminator to maximize the likelihood for the image-prediction pairs $x_l, \hat{y}_l$. Therefore, the adversarial prediction loss in S is

$$L_{G_{\text{pred}_{(x_l,\hat{y}_l)}}} = -\mathbb{E}_{x_l,\hat{y}_l \sim G} \log[1 - p(z_l = n+1 \mid (x_l, \hat{y}_l)]. \tag{3.40}$$

Similarly, for the unlabeled data, $x_u, \hat{y}_u$, the adversarial prediction loss is

$$L_{G_{\text{pred}_{(x_u,\hat{y}_u)}}} = -\mathbb{E}_{x_u,\hat{y}_u \sim G} \log[1 - p(z_u = n+1 \mid (x_u, \hat{y}_u)]. \tag{3.41}$$

Since the main objective of G is to generate the segmentation map, a small weight is used for the adversarial loss terms for both labeled and unlabeled data.

The class discriminator D is trained on multiple objectives—adversary on the segmentation mask generator G's output and classification of the images into the real or surrogate classes. Since the model is trained on both labeled and unlabeled training data, the loss function L_D of the class discriminator D includes both supervised and unsupervised losses. Function L_D includes five different loss terms: 1) supervised classification loss on $(\hat{c}_l \mid x_l, y_l)$, 2) self-supervised classification loss on unlabeled data $(\hat{c}_u \mid x_u, \hat{y}_u)$, 3) adversarial real loss on x_l, y_l, 4) adversarial prediction loss on $x_l, \hat{y}_l$, and 5) adversarial prediction loss on $x_u, \hat{y}_u$. The supervised losses are calculated as

$$L_{D_{\text{sup}}} = -\mathbb{E}_{x_l, y_l, c_l \sim p_{\mathcal{D}_\mathcal{L}}} \log \left[p(\hat{c}_l = (c_l = i) \mid x_l, y_l; i < n + 1) \right] \tag{3.42}$$

and

$$L_{D_{\text{self}}} = -\mathbb{E}_{x_u, \hat{y}_u, c_u \sim p_{\mathcal{D}_\mathcal{U}}} \log \left[p(\hat{c}_u = (c_u = i) \mid x_u, y_u; i < n + 1) \right]. \tag{3.43}$$

Now, for the labeled data $\mathcal{D}_L$, D can receive two-way inputs (x_l, y_l) or $(x_l, \hat{y}_l)$. Therefore, the adversarial losses for $\mathcal{D}_L$ are

$$L_{D_{\text{gt}}(\mathcal{D}_L)} = -\mathbb{E}_{x_l, y_l \sim p_{(\mathcal{D}_L)}} \log[1 - p(c_l = n + 1 \mid x_l, y_l)] \tag{3.44}$$

and

$$L_{D_{\text{pred}}(\mathcal{D}_L)} = -\mathbb{E}_{(x_l, \hat{y}_l) \sim S} \log[p(c_l = n + 1 \mid x_l, \hat{y}_l)]. \tag{3.45}$$

For the unlabeled data $\mathcal{D}_U$, we calculate only the adversarial prediction loss

$$L_{D_{\text{pred}}(\mathcal{D}_U)} = -\mathbb{E}_{(x_u, \hat{y}_u) \sim S} \log[p(c_u = n + 1 \mid x_u, \hat{y}_u)]. \tag{3.46}$$

CHAPTER 4

Experimental Evaluations

This chapter presents implementation details and experiments with the models developed in Chapter 3 and reports our results. The datasets employed are detailed in Appendix A.

4.1 3D Segmentation of Pulmonary Lobes

The segmentation of lung lobes from chest CT scans was validated using three datasets: LIDC, LTRC, and LOLA11.

4.1.1 Implementation Details

Baseline: For our baseline comparison, we used a U-Net architecture (Ronneberger et al., 2015a) and a dense V-Net (DV-Net). The former is used in the most recent published article for lung lobe segmentation (George et al., 2017) and the latter is a strong baseline for comparison, which we are the first to employ for lung lobe segmentation.

Training DV-Net and PDV-Net: The models were trained on the LIDC dataset; and tested on all three (LIDC, LTRC, and LOLA11) sets. For the DV-Net and our PDV-Net models, the training volumes were first normalized, followed by rescaling to $512 \times 512 \times 64$, using one NVIDIA Titan XP GPU. Due to the large memory footprint of the model, the gradient check-pointing method (Bulatov, 2018) was used for memory-efficient back-propagation. Additionally, batch-wise spatial dropout (Gibson et al., 2018) was incorporated for regularization purposes. The training was performed on a 64-bit Intel Xeon E5-2697 v4 2.30 GHz CPU system with 256 GB of RAM. We used the Adam

optimizer (Kingma and Ba, 2014) with a learning rate of 0.01 and a weight decay of $1e-7$.

Training U-Net: For the 2D U-Net model, the implemented architecture is symmetric and consists of four contracting and expanding layers, starting with 16 features in the first layer and doubling the number of features in each step. Each contracting layer consists of two 3×3 convolutions and a ReLU activation followed by a 2×2 max-pooling layer. The expansion path consists of an up-convolution with feature concatenation from the respective contracting layer, and two 3×3 convolutions. In addition, all the ReLU layers are preceded by a batch-normalization layer. To improve the training process, we also used a generalized Dice score as the loss function, such that the contribution of each class in the image to the gradients is balanced. We trained the network with axial slices from all the training volumes, each sized 512×512 and normalized to have values between 0 and 1. To avoid over-fitting to the background class, we used only the axial slices, wherein at least one lung lobe is present. We used the Adam optimizer with a learning rate of $5e-5$ and batches of 10 images.

4.1.2 LIDC Results

Table 4.1 shows the calculated overall and lobe-wise Dice scores and standard deviations for each of the models. Our PDV-Net model, with an overall score of 0.939 ± 0.020, significantly outperformed the 2D model and yielded consistently larger Dice scores for each of the lung lobes against both the DV-Net and U-Net. Moreover, the lower standard deviation for each lobe indicates that our progressive model is more robust. Figure 4.1 provides a qualitative comparison between the three models, showing that our PDV-Net model captures lung fissures better than the 2D U-Net and DV-Net. The superiority of our PDV-Net model is evident both in slice (axial, coronal, sagittal) and 3D views.

We further used Bland-Altman plots to measure the agreement between our PDV-Net and ground truth segmentations of the 84 LIDC cases (Figure 4.2). Good agreement was observed between our segmentation model and ground truth in every plot (Lung and

Table 4.1: Performance comparison of our 3D progressive dense V-Net against the 2D U-Net and 3D dense V-Net models in segmenting 84 LIDC and 154 LTRC cases. Mean Dice score and standard deviation for each of the five lobes are reported.

Dataset	Model	RUL	RML	RLL	LUL	LLL	Overall
LIDC(84)	2D U-Net	0.908 ± 0.049	0.844 ± 0.076	0.940 ± 0.054	0.959 ± 0.042	0.949 ± 0.056	0.920 ± 0.043
	3D DV-Net	0.929 ± 0.036	0.873 ± 0.058	0.951 ± 0.018	0.958 ± 0.020	0.949 ± 0.041	0.932 ± 0.023
	3D PDV-Net	**0.937 ± 0.031**	**0.882 ± 0.057**	**0.956 ± 0.017**	**0.966 ± 0.014**	**0.966 ± 0.037**	**0.939 ± 0.020**
LTRC(154)	2D U-Net	0.914 ± 0.039	0.866 ± 0.054	0.952 ± 0.023	0.961 ± 0.023	0.954 ± 0.021	0.929 ± 0.025
	3D DV-Net	0.949 ± 0.013	0.901 ± 0.021	0.959 ± 0.009	0.961 ± 0.007	0.958 ± 0.012	0.946 ± 0.008
	3D PDV-Net	**0.952 ± 0.011**	**0.908 ± 0.020**	**0.961 ± 0.008**	**0.966 ± 0.006**	**0.960 ± 0.010**	**0.950 ± 0.007**

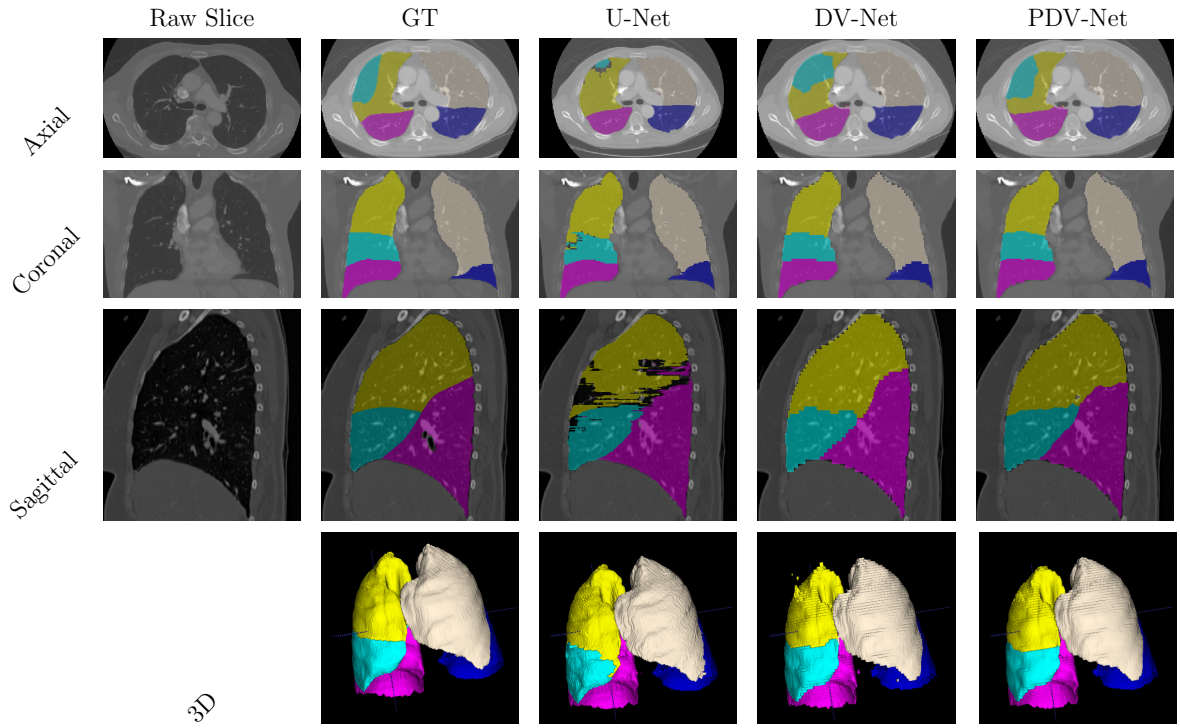


Figure 4.1: Qualitative comparison of PDV-Net's superior performance, both in slice and volume level, against DV-Net and U-Net. Note how noisy patches and rough boundaries are removed from the final segmentation generated by the PDV-Net. Color coding: almond: LUL, blue: LLL, yellow: RUL, cyan: RML, pink: RLL.

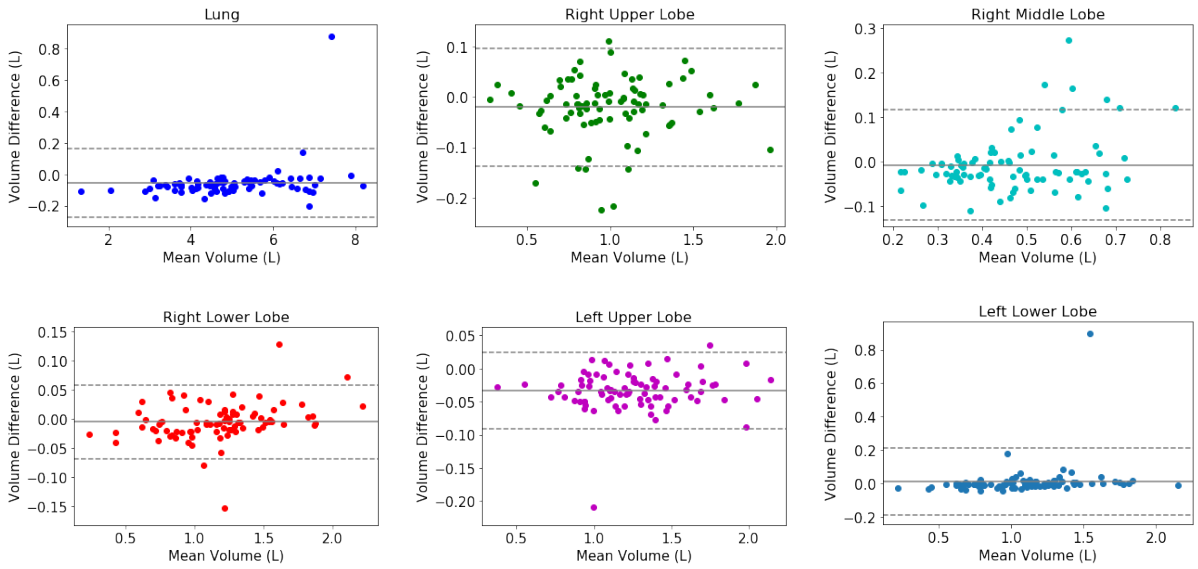


Figure 4.2: Bland-Altman plots show the agreement between our 3D PDV-Net and ground truth.

LLL being the two best agreements). Pearson correlation showed that all six volume sets in ground truth are strongly correlated with the corresponding six volume sets in the PDV-Net segmentation, with $p < 0.001$.

4.1.3 LTRC Results

Table 4.1 shows that the 3D progressive dense V-Net achieves an average Dice score of 0.950 ± 0.007, significantly improving the dense V-Net (0.946 ± 0.008). Once again, the progressive dense V-Net model outperformed the 2D U-Net model with an average Dice score of 0.929 ± 0.025. Individual lobes were segmented better by our proposed 3D progressive dense V-Net model than by the 3D dense V-Net and the 2D U-Net models (Table 4.1). Note that the LTRC dataset includes many pathological cases where the fissure lines are either invisible, distorted, or absent in the presence of pathologies such as emphysema, fibrosis, etc. As a result, lobe segmentation becomes far more challenging. Nevertheless, our model performed well in segmenting lobes in pathological cases from the LTRC dataset. Moreover, our model outperformed the model of George et al. (2017) in segmenting the LTRC cases both in Dice score (0.941 ± 0.255) and inference speed (4-8 minutes per case).

Table 4.2: Performance evaluation of our 3D PDV-Net model on 55 LOLA cases, showing lobe-wise mean Dice scores, standard deviations, median scores, first quartiles, and third quartiles

Lobe	Mean ± SD	Q_1	Median	Q_3
RUL	0.9518 ± 0.1750	0.9371	0.9688	0.9881
RML	0.8621 ± 0.4149	0.8107	0.9284	0.9663
RLL	0.9581 ± 0.1993	0.9621	0.9829	0.9881
LUL	0.9551 ± 0.2160	0.9644	0.9834	0.9924
LLL	0.9342 ± 0.3733	0.9546	0.9805	0.9902
Overall	0.9345			
(Giuliani et al., 2018)	0.9282			
(Bragman et al., 2017)	0.9384			
(van Rikxoort et al., 2010)	0.9195			

Jaccard score to Dice score conversion: Dice $= 2 \times$ Jaccard$/(1 +$ Jaccard$)$

4.1.4 LOLA11 Results

Our segmentation results for the LOLA11 cases were evaluated by the organizers of LOLA11. To be consistent with our previous analyses, the Jaccard scores computed by the organizers were converted to Dice scores. The results are shown in Table 4.2. Our method achieved an overall Dice score of 0.934, which is very competitive to the state-of-the-art reliant method (Bragman et al., 2017) with a Dice score of 0.938 (reliant approach), while outperforming the methods of Giuliani et al. (2018) and van Rikxoort et al. (2010).

Figure 4.3 shows the segmentation results for the LOLA11 cases. For the left lung in Case 8, the LUL and LLL Dice scores were 0.9940 and 0.9926, respectively. For the right lung in Case 6, the scores are as follows: RUL: 0.9580, RML: 0.9480, and RLL: 0.9869. Again, for the left lung of Case 21, the segmentation Dice scores were relatively low. For the left lung in Case 21, the LUL score was 0.8170 and the LLL score was 0.3035. For the right lung in Case 55, although the right lower lobe was segmented with a high Dice score of 0.9818, because of the invisibility of the horizontal fissure, the RUL and RML had low segmentation Dice scores of 0.6827 and 0.7499, respectively.

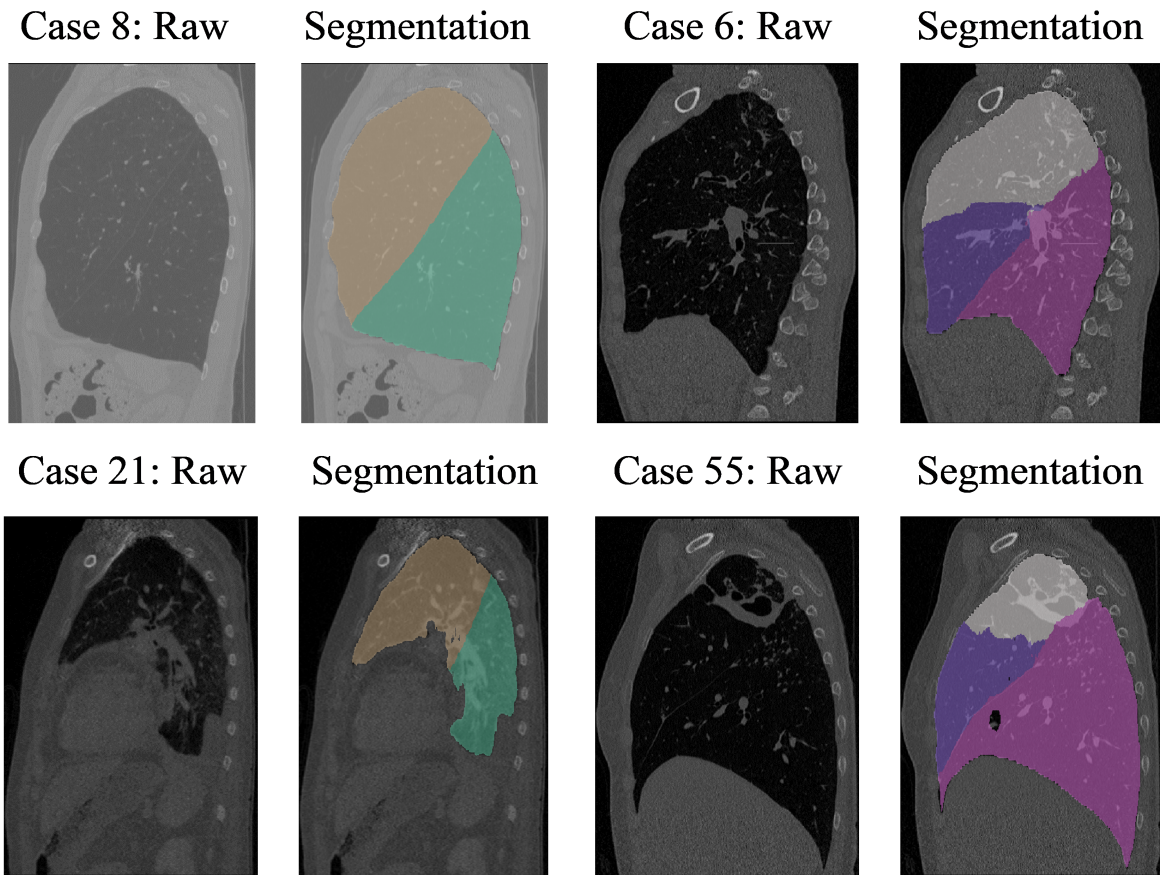


Figure 4.3: Sagittal plane visualization of LOLA11 segmentation by our 3D PDV-Net: good cases (upper row) and failure cases (bottom row)

4.1.5 Robustness Analysis

We further investigated the robustness of our model by grouping the 84 LIDC cases in three ways. For the first grouping, the Dice scores were put in three different Z-spacing buckets: Z-spacing ≤ 1, $1 <$ Z-spacing < 2, and Z-spacing ≥ 2. In the second grouping, the Dice scores were put in four manufacturer buckets: GE, Philips, Siemens, and Toshiba. In the third grouping, the Dice scores were grouped according to the reconstruction kernel into 3 buckets: soft, lung, and bone. A one-way ANOVA analysis confirmed that there were no significant differences (p-value < 0.05) between the average Dice scores of the buckets within each grouping, suggesting that our model is robust against the choice of reconstruction kernel, size of reconstruction interval, and different CT scanner vendors.

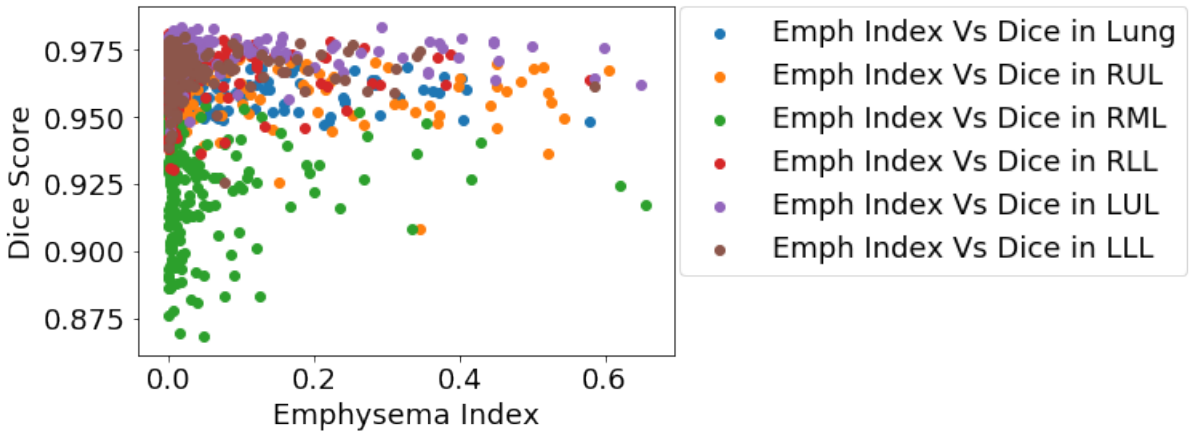


Figure 4.4: Plots of lobe-wise and overall segmentation accuracy (Dice scores) of our model versus the emphysema indices of the LTRC test cases reveal insignificant correlation.

Moreover, nodule volume in each of the 84 cases does not affect the lobe segmentation performance. There is no correlation between nodule volume and lobe segmentation accuracy, found from Pearson correlation (p-value < 0.05).

We also studied how the segmentation correlation is affected by lung pathologies, by analyzing the correlation between Dice scores and the emphysema index; i.e., the proportion of the lungs affected by emphysema (in the range 0–1). For the LTRC cases, we associated lobe-wise emphysema indices by calculating the proportion of emphysema voxels (voxels marked as emphysema in the LTRC ground truth) in each of the lobes, as well as overall emphysema indices for both lungs. Figure 4.4 shows plots of the per-lobe and overall emphysema indices versus segmentation performance. The small Pearson correlation (p-value < 0.05) reveals that the lobe segmentation accuracy is uncorrelated with the emphysema index, confirming the robustness of our model in segmenting lobes in challenging cases with clear evidence of emphysema.

4.1.6 Speed Analysis

Our 3D PDV-Net model takes approximately 2 seconds to segment lung lobes from one CT scan using a single Nvidia Titan XP GPU, which is 6x faster than the 2D U-Net model and at least 120x faster than the 2D U-Net followed by random walker-based approach

(George et al., 2017). To our knowledge from the lung lobe segmentation models reported in literature, ours is by far the fastest model. Note that no prior published research considered a 3D CNN model for lung lobe segmentation.

4.2 2D Segmentation & Analysis of Scoliosis

For the segmentation of vertebrae and measurement of scoliosis from spine X-ray images, we used the APSeg dataset.

4.2.1 Implementation Details

Models: As baselines, we used a regular U-Net model with binary cross-entropy (XE) and Dice loss functions. Moreover, to verify the robustness of our model, we experimented with training data augmentation (varying rotation, flipping, contrast, Gaussian noise, etc.). For simplicity, we refer to the baseline model as UD (UNet with Dice loss) and UX (UNet with XE loss), and to our model as PUD (Progressive UNet with Dice loss), PUX (Progressive UNet with XE loss), PUDA (Progressive UNet with Dice loss and data augmentation), and PUXA (Progressive UNet with XE loss and data augmentation).

Training: The models were trained on the training set while their performances were evaluated on the testing set. The validation set was used for hyper-parameter tuning and model selection.

Inputs: All the images were resized and normalized to $1024 \times 512 \times 1$ before feeding them to the network.

Hyperparameters: We used the Adam optimizer with adaptive learning rate starting with an initial rate of 0.01 and decreasing 10 times after every 20 epochs. A dropout of 0.25 was applied after every convolution layer except the ones for generating side-outputs and the final output.

Machine Configuration: We implemented Algorithm 1 in TensorFlow running on a Tesla P40 GPU in a system with a 64-bit Intel(R) Xeon(R) 440G CPU.

Segmentation Evaluation: For segmentation evaluation, along with qualitative visualization of masks and edges, we use the Dice index (DI), structural similarity index (SSIM), average Hausdorff distance (HD), and F1 score (F1).

Scoliosis Evaluation: For the evaluation of scoliosis, we calculated the Cobb angles, determined the indices of most tilted upper and lower vertebrae, and severity classification. Since the expert annotations include only the vertebrae labels for segmentation reference, we followed the same scoliosis measurement procedure for both the reference measurements and for our proposed PU-Net model.

4.2.2 Segmentation Results

Experimental results based on both qualitative and quantitative evaluations confirm the superiority of our model, which consistently provides improved segmentation with varying choice of loss functions (Dice loss and XE loss). Visualizations of the segmented vertebrae (Figure 4.5 and Figure 4.6) depict better distinctions of the individual vertebrae merely with binary segmentation. In all four quantitative measures, our models achieve better scores than the baseline models (Table 4.3). The superiority of our models is further confirmed by the whisker-box plots in Figure 4.7 and Bland-Altman plots in Figure 4.8.

Moreover, our end-to-end vertebrae segmentation achieves a better Dice similarity score than the recently published patch-wise segmentation method (Horng et al., 2019) (0.993 vs 0.952). While superior DI and F1 justifies the progressive addition of the side-outputs in pixel-wise predictions, better SSIM and HD depict the model's ability to learn the intrinsic shape and structure of the segmented vertebrae.

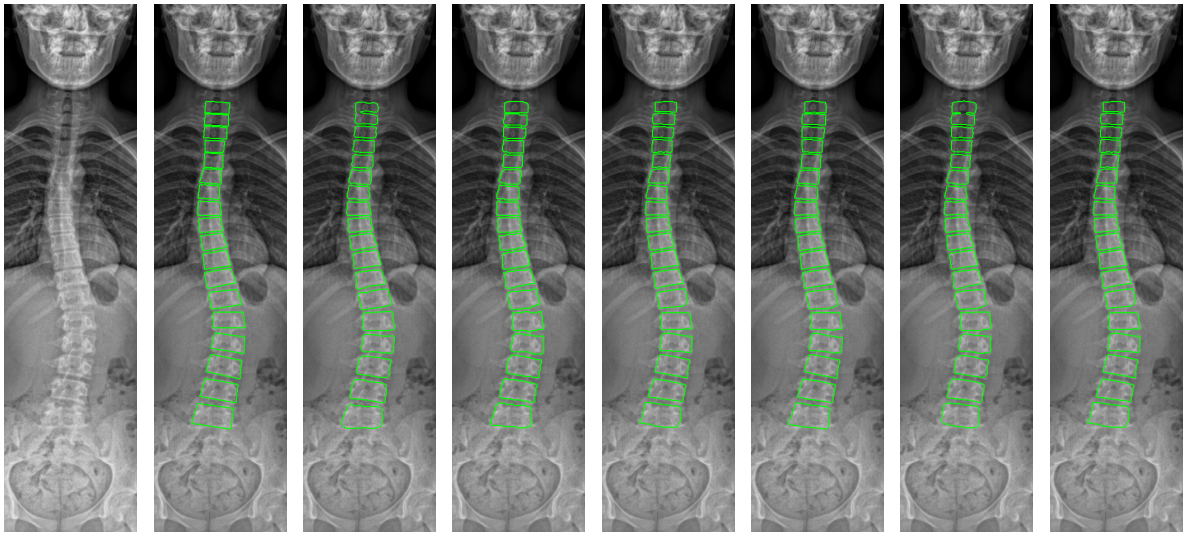


Figure 4.5: Boundary visualization of the predicted vertebrae masks in a spine X-Ray shows consistent improvement by our model over all other models.

Table 4.3: Performance comparison of the vertebrae segmentation models

Model	DI	SSIM	HD	F1
UD	0.970	0.961	5.246	0.896
UX	0.956	0.955	6.767	0.868
PUD	**0.993**	0.966	**4.597**	0.919
PUX	**0.993**	**0.970**	4.677	**0.922**
PUDA	0.992	0.966	**4.181**	0.913
PUXA	0.992	0.967	4.956	0.911

4.2.3 Scoliosis Results

For the evaluation of scoliosis, we compare the performance of our PUX model-based measurement against the reference measurement obtained by processing the expert's annotations. Figure 4.9 visualizes the Cobb angle results from vertebra masks of two sample X-rays. As reported in Table 4.4, our segmentation-based pipeline achieves very accurate Cobb angles. Good agreement is observed between our model and the reference measurement in each of the X-Rays in the test set with a mean angle difference of just 2.41 degrees, which is well below the acceptable error limit recommended by the experts (Cassar-Pullicino and Eisenstein, 2002). Comparing with some of the existing Cobb angle measurement techniques, our method achieves lower measurement error than those

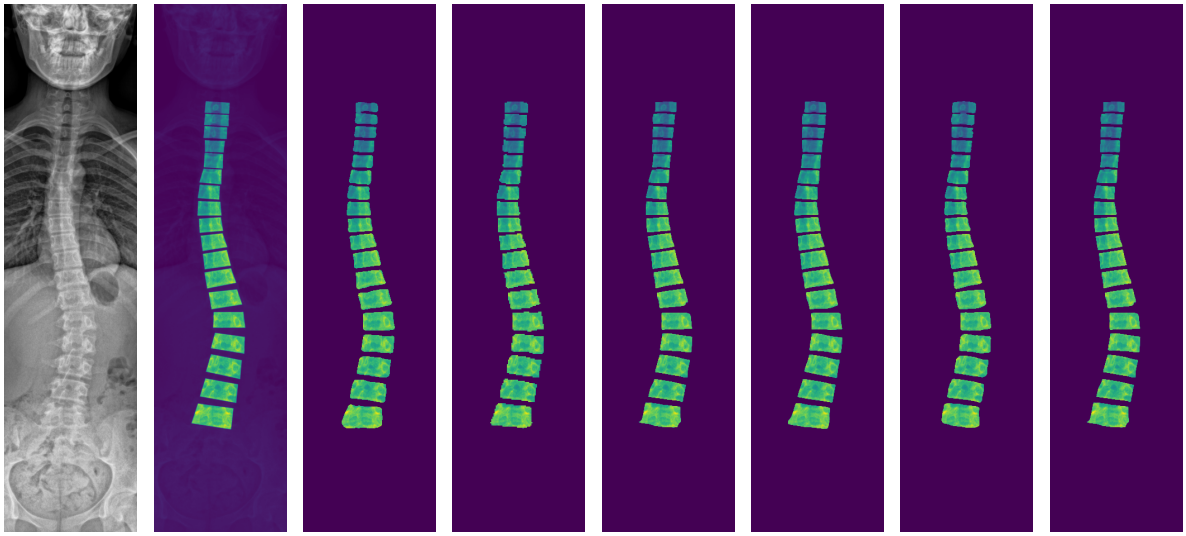


Figure 4.6: Visualization (zoomed) of the predicted vertebrae masks in a spine X-Ray shows consistent improvement by our model over all other models.

reported by Kusuma (2017) and Horng et al. (2019). Moreover, the categorization of scoliosis (Chowanska et al., 2012) indicates 100% diagnostic accuracy of our approach relative to the reference.

4.3 Semi-Supervised Simultaneous Image Generation and Classification

Two natural image datasets (SVHN, CIFAR-10) and two medical image datasets (CXR, SLC) were used for evaluating the semi-supervised joint image generation and classification performance.

4.3.1 Implementation Details

To compare the image generation and multiclass classification performance of our MAVEN model, we used two baselines, the Deep Convolutional GAN (DC-GAN) (Radford et al., 2015) and the VAE-GAN. The same generator and discriminator architectures were used for DC-GAN and MAVEN models and the same encoder was used for the VAE-GAN and MAVEN models. For our MAVENs, we experimented with 2, 3, and 5 discriminators. In

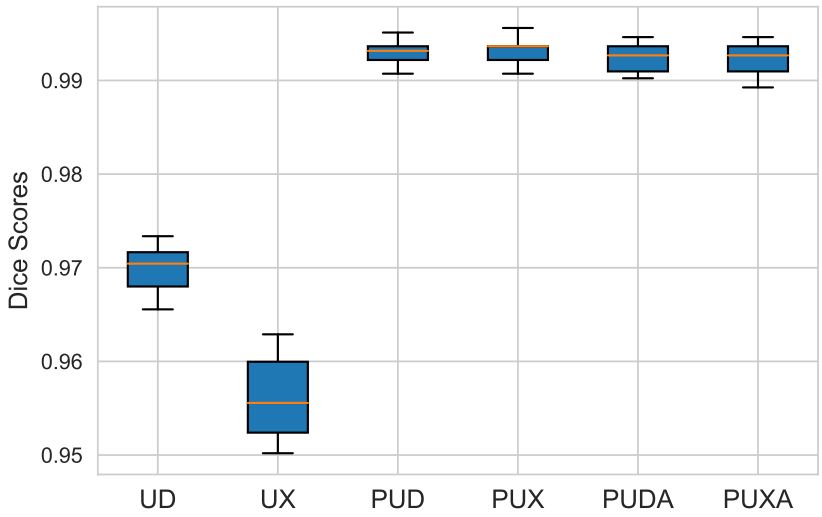


Figure 4.7: Whisker-Box plots of all four models showing consistent performance of our model with varying losses in segmenting 18 scoliosis-relevant vertebrae from the spine X-ray test set.

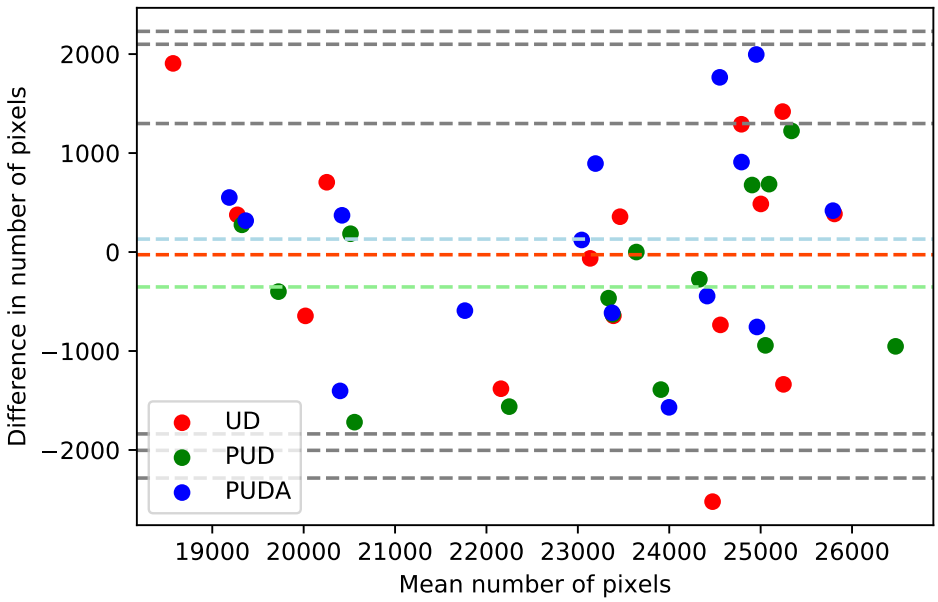


Figure 4.8: Bland-Altman plots show better agreement of our model relative to competing models with Dice loss in segmenting 18 vertebrae from the scoliotic X-Ray images in the test set.

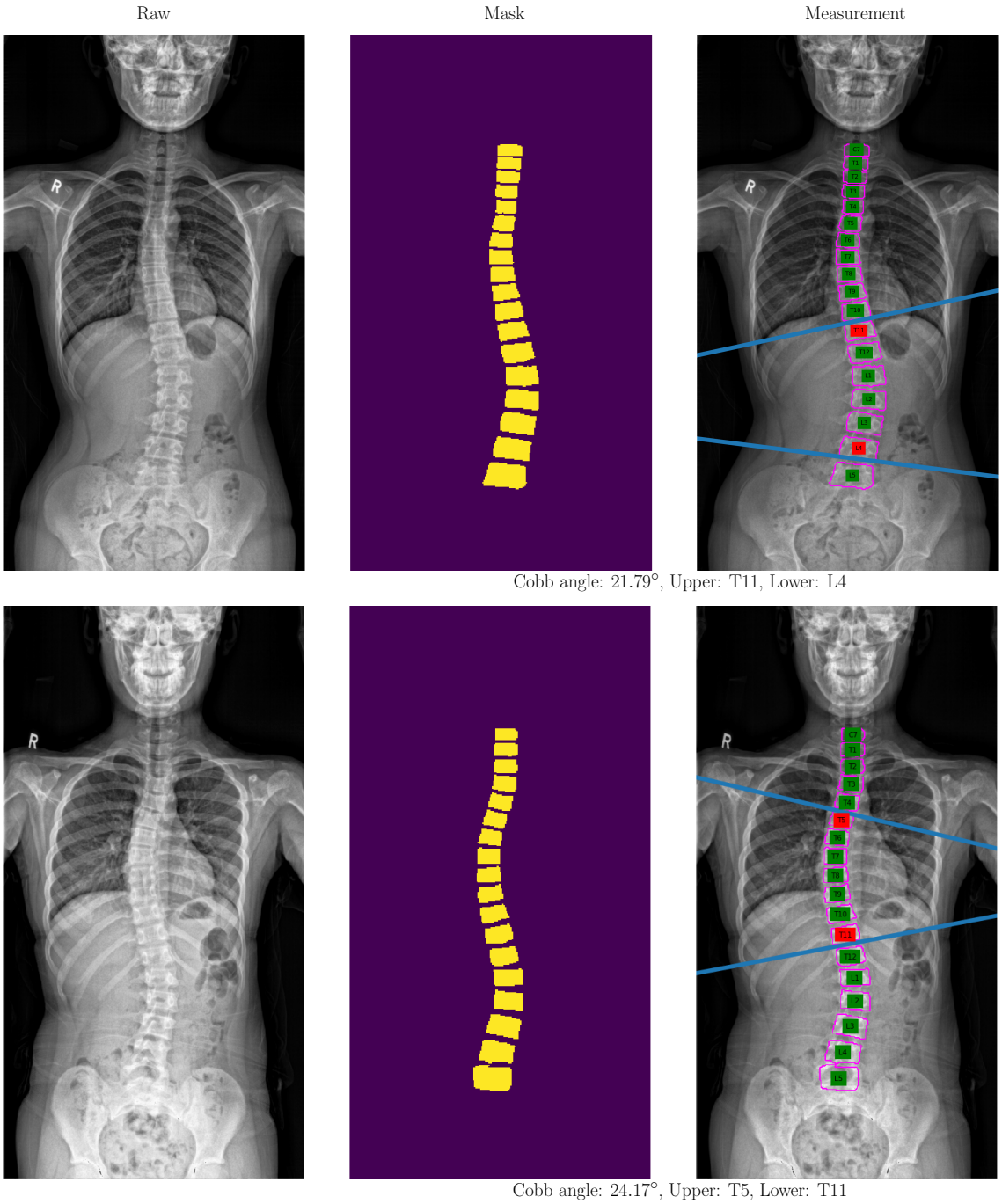


Figure 4.9: From input X-ray image (left), to segmentation mask prediction (middle), to vertebrae identification and scoliosis measurement (right) in our pipeline.

Table 4.4: Performance of our method for calculating Cobb angle and scoliosis severity in the test set relative to reference measurements

Test	Reference Measurement				Prediction of our PUX Model				$\lvert Deviation \rvert$
ID	Upper_vert	Lower_vert	Cobb_angle	Severity	Upper_vert	Lower_vert	Cobb_angle	Severity	
1	T10	L3	21.92°	mild	T10	L3	19.26°	mild	2.66
2	T12	L4	09.52°	normal	T6	L3	04.75°	normal	4.77
3	T11	L3	13.88°	mild	T11	L3	14.48°	mild	0.6
4	T5	T10	18.78°	mild	T5	T11	16.16°	mild	2.62
5	T6	T10	20.53°	mild	T10	L4	20.22°	mild	0.31
6	T11	L4	20.38°	mild	T11	L4	23.35°	mild	2.97
7	T10	L2	40.71°	moderate	T11	L3	41.38°	moderate	0.67
8	T6	T12	23.96°	mild	T6	T12	20.93°	mild	3.03
9	T5	T11	21.07°	mild	T6	T11	23.22°	mild	2.15
10	T6	T10	14.81°	mild	T1	L3	16.10°	mild	1.29
11	T12	L4	31.94°	moderate	T12	L4	28.60°	moderate	3.34
12	T10	L1	24.82°	mild	T9	L1	18.92°	mild	5.92
13	T12	L3	15.69°	mild	T10	L3	14.79°	mild	0.90
14	T12	L4	24.25°	mild	T8	T12	22.72°	mild	1.53
15	T12	L3	21.02°	mild	T11	L3	18.61°	mild	2.41

addition to using the mean feedback of the multiple discriminators, we also experimented with feedback from a randomly selected discriminator. The six MAVEN variants are therefore denoted MAVEN-m2D, MAVEN-m3D, MAVEN-m5D, MAVEN-r2D, MAVEN-r3D, and MAVEN-r5D, where "m" indicates mean feedback while "r" indicates random feedback.

All the models were implemented in TensorFlow and run on a single Nvidia Titan GTX (12GB) GPU. For the discriminator, after every convolutional layer, a dropout layer was added with a dropout rate of 0.4. For all the models, we consistently used the Adam optimizer with a learning rate of 2.0^{-4} for G and D, and 1.0^{-5} for E, with a momentum of 0.9. All the convolutional layers were followed by batch normalizations. Leaky ReLU activations were used with $\alpha = 0.2$.

4.3.2 Evaluation

4.3.2.1 Image Generation Performance

There are no perfect performance metrics for measuring the quality of generated samples. However, to assess the quality of the generated images, we employed the widely used Fréchet Inception Distance (FID) (Heusel et al., 2017) and a simplified version of the

Descriptive Distribution Distance (DDD) (Imran et al., 2017). To measure the Fréchet distance between two multivariate Gaussians, the generated samples and real data samples are compared through their distribution statistics:

$$\mathrm{FID} = \|\mu_{\mathrm{data}} - \mu_{\mathrm{syn}}\|^2 + \mathrm{Tr}\left(\Sigma_{\mathrm{data}} + \Sigma_{\mathrm{syn}} - 2\sqrt{\Sigma_{\mathrm{data}}\Sigma_{\mathrm{syn}}}\right). \tag{4.1}$$

Two distribution samples, $\mathcal{X}_{\mathrm{data}} \sim \mathcal{N}(\mu_{\mathrm{data}}, \Sigma_{\mathrm{data}})$ and $\mathcal{X}_{\mathrm{syn}} \sim \mathcal{N}(\mu_{\mathrm{syn}}, \Sigma_{\mathrm{syn}})$, for real and model data, respectively, are calculated from the 2,048-dimensional activations of the pool3 layer of Inception-v3 (Salimans et al., 2016). DDD measures the closeness of a generated data distribution to a real data distribution by comparing descriptive parameters from the two distributions. We propose a simplified version based on the first four moments of the distributions, computed as the weighted sum of normalized differences of moments, as follows:

$$\mathrm{DDD} = -\sum_{i=1}^{4} \log w_i \left|\mu_{\mathrm{data}_i} - \mu_{\mathrm{syn}_i}\right|, \tag{4.2}$$

where the μ_{data_i} are the moments of the data distribution, the μ_{syn_i} are the moments of the model distribution, and the w_i are the corresponding weights found in an exhaustive search. The higher-order moments are weighted more in order to emphasize the stability of a distribution. For both FID and DDD, lower scores are better.

4.3.2.2 Image Classification Performance

To evaluate model performance in classification, we used two measures, image-level classification accuracy and class-wise F1 scoring. The F1 score is

$$\mathrm{F1} = \frac{2 \times \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}, \tag{4.3}$$

with

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \text{and} \qquad \mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{4.4}$$

Table 4.5: Minimum FID and DDD scores achieved by the DC-GAN, VAE-GAN, and MAVEN models for the two natural image datasets (CIFAR-10, SVHN).

CIFAR-10			SVHN		
Model	FID	DDD	Model	FID	DDD
DC-GAN	61.293±0.209	0.265	DC-GAN	16.789±0.303	0.343
VAE-GAN	15.511±0.125	0.224	VAE-GAN	13.252±0.001	0.329
MAVEN-m2D	12.743±0.242	0.223	MAVEN-m2D	11.675±0.001	0.309
MAVEN-m3D	**11.316±0.808**	**0.190**	MAVEN-m3D	11.515±0.065	0.300
MAVEN-m5D	12.123±0.140	0.207	MAVEN-m5D	10.909±0.001	**0.294**
MAVEN-r2D	12.820±0.584	0.194	MAVEN-r2D	11.384±0.001	0.316
MAVEN-r3D	12.620±0.001	0.202	MAVEN-r3D	**10.791±0.029**	0.357
MAVEN-r5D	18.509±0.001	0.215	MAVEN-r5D	11.052±0.751	0.323
DO-GAN (Mordido et al., 2018)	88.60±0.08	-			
TTUR (Heusel et al., 2017)	36.9	-			
C-GAN (Unterthiner et al., 2017)	27.300	-			
AIQN (Ostrovski et al., 2018)	49.500	-			
SN-GAN (Miyato et al., 2018)	21.700	-			
LM (Ravuri et al., 2018)	18.9	-			

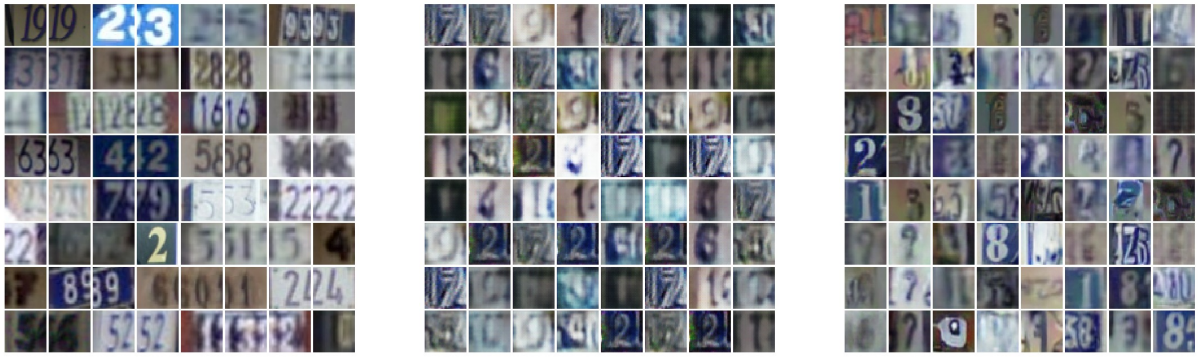
Table 4.6: Minimum FID and DDD scores achieved by the DC-GAN, VAE-GAN, and MAVEN models for the two medical image datasets (CXR, SLC).

CXR			SLC		
Model	FID	DDD	Model	FID	DDD
DC-GAN	152.511±0.370	0.145	DC-GAN	1.828±0.370	0.795
VAE-GAN	141.422±0.580	0.107	VAE-GAN	1.828±0.580	0.795
MAVEN-m2D	141.339±0.420	0.138	MAVEN-m2D	1.874±0.270	0.802
MAVEN-m3D	**140.865±0.983**	**0.018**	MAVEN-m3D	**0.304±0.018**	**0.249**
MAVEN-m5D	147.316±1.169	0.100	MAVEN-m5D	1.518±0.190	0.793
MAVEN-r2D	154.501±0.345	0.038	MAVEN-r2D	1.505±0.130	0.789
MAVEN-r3D	158.749±0.297	0.179	MAVEN-r3D	0.336±0.080	0.783
MAVEN-r5D	152.778±1.254	0.180	MAVEN-r5D	1.812±0.014	0.795

where TP, FP, and FN are the number of true positives, false positives, and false negatives, respectively.

4.3.3 Results

We measured the image classification performances of the models with cross-validation and in the following sections report the average scores from running each model 10 times.



(a) Real samples (b) DC-GAN (c) VAE-GAN

(d) MAVEN-m2D (e) MAVEN-m3D (f) MAVEN-m5D

(g) MAVEN-r2D (h) MAVEN-r3D (i) MAVEN-r5D

Figure 4.10: Visual comparison of image samples from the SVHN dataset against those generated by the different models.

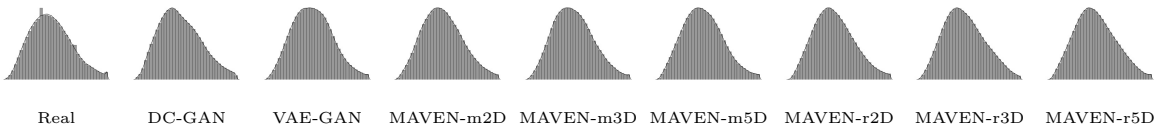


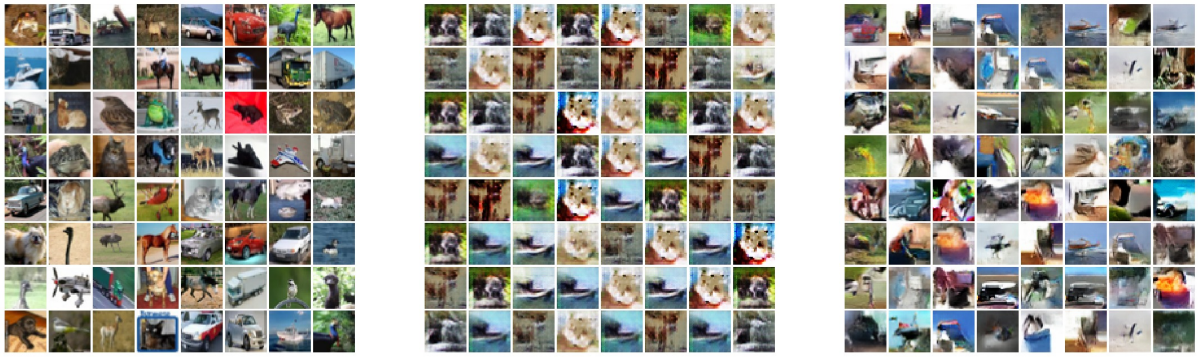
Figure 4.11: Histograms of the real SVHN training data, and of the data generated by the DC-GAN and VAE-GAN models and by our MAVEN models with mean and random feedback from 2, 3, and 5 discriminators.

4.3.3.1 SVHN

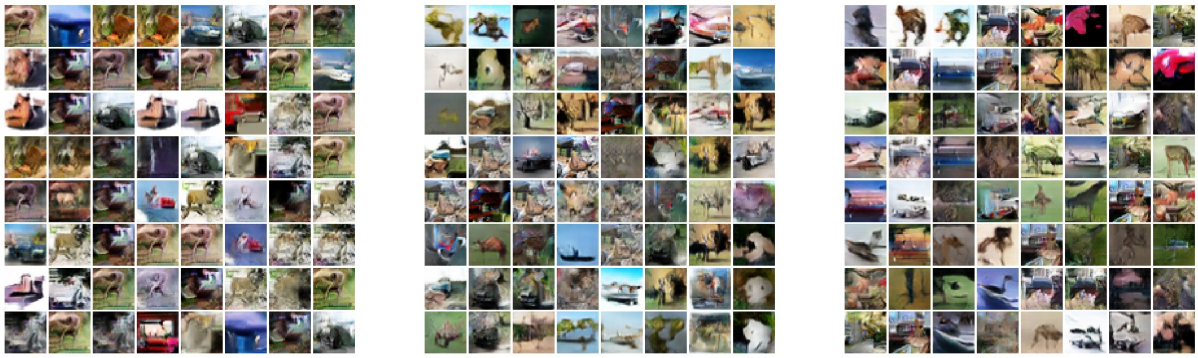
For the SVHN dataset, we randomly selected 7,326 labeled images and they along with the remaining 65,931 unlabeled images were provided to the network as training data. All the models were trained for 300 epochs and then evaluated. We generated new images equal in number to the training set size. Figure 4.10 presents a visual comparison of a random selection of images generated by the DC-GAN, VAE-GAN, and MAVEN models and real training images. Figure 4.11 compares the image intensity histograms of 10K randomly sampled real images and equally many images sampled from among those generated by each of the different models.

Generally speaking, our MAVEN models generate images that are more realistic than those generated by the DC-GAN and VAE-GAN models. This was further corroborated by randomly sampling 10K generated images and 10K real images. The generated image quality measurement was performed for the eight different models. Table 4.5 reports the resulting FID and DDD scores. For the FID score calculation, the score is reported after running the pre-trained Inception-v3 network for 20 epochs for each model. The MAVEN-r3D model achieved the best FID score and the best DDD score was achieved by the MAVEN-m5D model.

Table 4.7 compares the classification performance of all the models for the SVHN dataset. The MAVEN model consistently outperformed the DC-GAN and VAE-GAN classifiers both in classification accuracy and class-wise F1 scores. Among all the models, our MAVEN-m2D and MAVEN-m3D models were the most accurate.



(a) Real samples (b) DC-GAN (c) VAE-GAN



(d) MAVEN-m2D (e) MAVEN-m3D (f) MAVEN-m5D

(g) MAVEN-r2D (h) MAVEN-r3D (i) MAVEN-r5D

Figure 4.12: Visual comparison of image samples from the CIFAR-10 dataset against those generated by the different models.

Table 4.7: Average cross-validation accuracy and class-wise F1 scores in the semi-supervised classification performance comparison of the DC-GAN, VAE-GAN, and MAVEN models using the SVHN dataset.

Model	Accuracy	F1 scores									
		0	1	2	3	4	5	6	7	8	9
DC-GAN	0.876	0.860	0.920	0.890	0.840	0.890	0.870	0.830	0.890	0.820	0.840
VAE-GAN	0.901	0.900	0.940	0.930	0.860	0.920	0.900	0.860	0.910	0.840	0.850
MAVEN-m2D	**0.909**	0.890	0.930	0.940	0.890	0.930	0.900	0.870	0.910	0.870	0.890
MAVEN-m3D	**0.909**	0.910	0.940	0.940	0.870	0.920	0.890	0.870	0.920	0.870	0.860
MAVEN-m5D	0.905	0.910	0.930	0.930	0.870	0.930	0.900	0.860	0.910	0.860	0.870
MAVEN-r2D	0.905	0.910	0.930	0.940	0.870	0.930	0.890	0.860	0.920	0.850	0.860
MAVEN-r3D	0.907	0.890	0.910	0.920	0.870	0.900	0.870	0.860	0.900	0.870	0.890
MAVEN-r5D	0.903	0.910	0.930	0.940	0.860	0.910	0.890	0.870	0.920	0.850	0.870

4.3.3.2 CIFAR-10

For the CIFAR-10 dataset, we used 50K training images, only 5K of them labeled. All the models were trained for 300 epochs and then evaluated. We generated new images equal in number to the training set size. Figure 4.12 visually compares a random selection of images generated by the DC-GAN, VAE-GAN, and MAVEN models and real training images. Figure 4.13 compares the image intensity histograms of 10K randomly sampled real images and equally many images sampled from among those generated by each of the different models. As the tabulated results in Table 4.5 suggest, our MAVEN models achieved better FID scores than some of the recently published models. Note that those models were implemented in different settings.

As for the visual comparison, the FID and DDD scores confirmed more realistic image generation by our MAVEN models compared to the DC-GAN and VAE-GAN models. The MAVEN models have smaller FID scores, except for MAVEN-r5D. MAVEN-m3D has the smallest FID and DDD scores among all the models.

Table 4.8 compares the classification performance of all the models with the CIFAR-10 dataset. All the MAVEN models performed better than the DC-GAN and VAE-GAN models. In particular, MAVEN-m5D achieved the best classification accuracy and F1 scores.

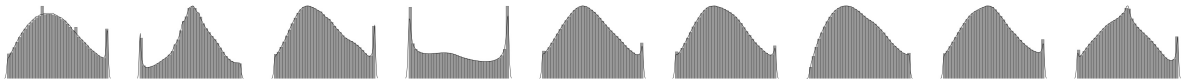


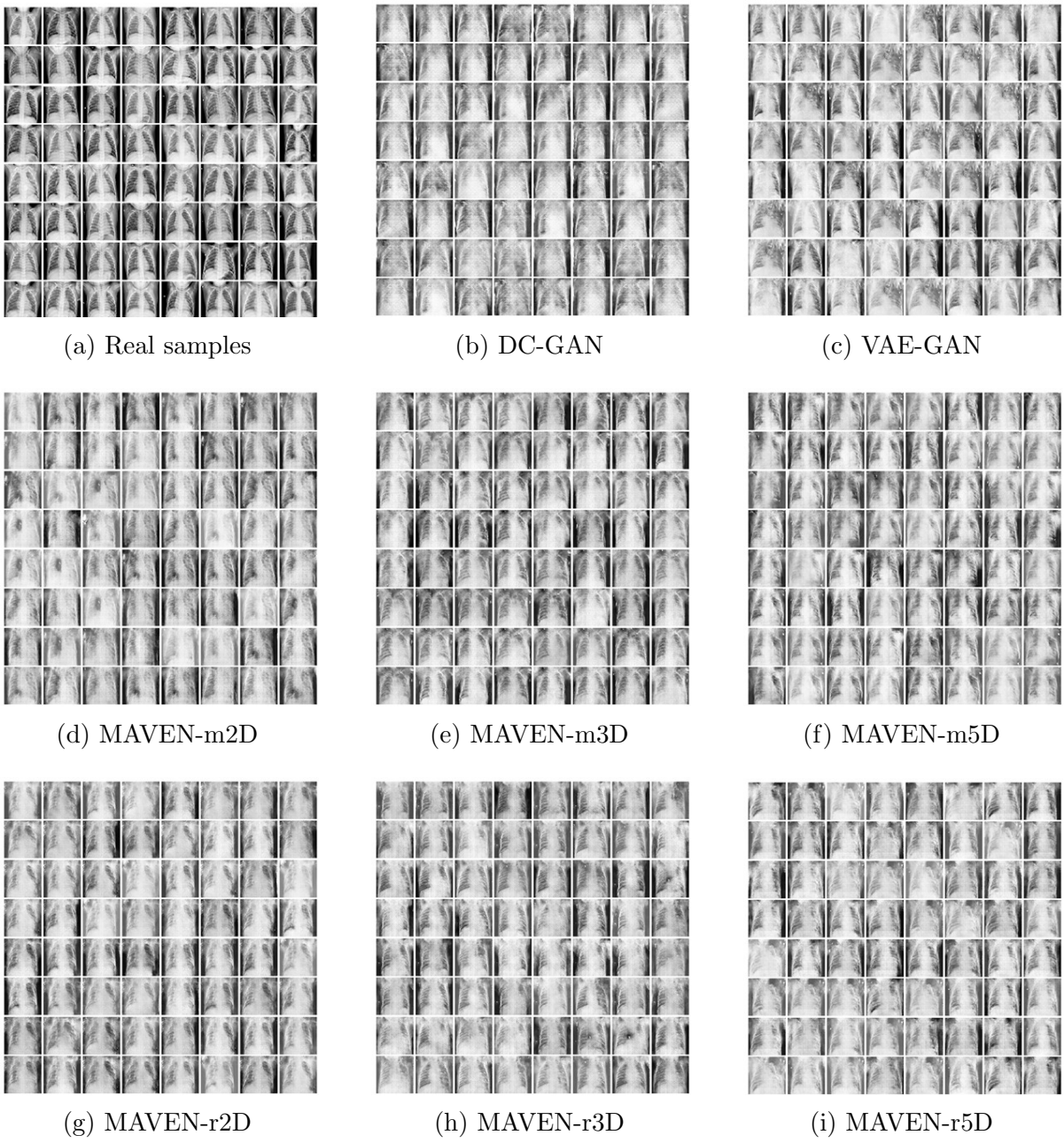
Figure 4.13: Histograms of the real CIFAR-10 training data, and of the data generated by the DC-GAN and VAE-GAN models and by our MAVEN models with mean and random feedback from 2, 3, and 5 discriminators.

Table 4.8: Average cross-validation accuracy and class-wise F1 scores in the semi-supervised classification performance comparison of the DC-GAN, VAE-GAN, and MAVEN models using the CIFAR-10 dataset.

Model	Accuracy	F1 scores									
		plane	auto	bird	cat	deer	dog	frog	horse	ship	truck
DC-GAN	0.713	0.760	0.840	0.560	0.510	0.660	0.590	0.780	0.780	0.810	0.810
VAE-GAN	0.743	0.770	0.850	0.640	0.560	0.690	0.620	0.820	0.770	0.860	0.830
MAVEN-m2D	0.761	0.800	0.860	0.650	0.590	0.750	0.680	0.810	0.780	0.850	0.850
MAVEN-m3D	0.759	0.770	0.860	0.670	0.580	0.700	0.690	0.800	0.810	0.870	0.830
MAVEN-m5D	**0.771**	0.800	0.860	0.650	0.610	0.710	0.640	0.810	0.790	0.880	0.820
MAVEN-r2D	0.757	0.780	0.860	0.650	0.530	0.720	0.650	0.810	0.800	0.870	0.860
MAVEN-r3D	0.756	0.780	0.860	0.640	0.580	0.720	0.650	0.830	0.800	0.870	0.830
MAVEN-r5D	0.762	0.810	0.850	0.680	0.600	0.720	0.660	0.840	0.800	0.850	0.820

Table 4.9: Average cross-validation accuracy and class-wise F1 scores for the semi-supervised classification performance comparison of the DC-GAN, VAE-GAN, and MAVEN models using the CXR dataset.

Model	Accuracy	F1 scores		
		Normal	b-Pneumonia	v-Pneumonia
DC-GAN	0.461	0.300	0.520	0.480
VAE-GAN	0.467	0.220	0.640	0.300
MAVEN-m2D	0.469	0.310	0.620	0.260
MAVEN-m3D	**0.525**	0.640	0.480	0.480
MAVEN-m5D	0.477	0.380	0.480	0.540
MAVEN-r2D	0.478	0.280	0.630	0.310
MAVEN-r3D	0.506	0.440	0.630	0.220
MAVEN-r5D	0.483	0.170	0.640	0.240



(a) Real samples (b) DC-GAN (c) VAE-GAN

(d) MAVEN-m2D (e) MAVEN-m3D (f) MAVEN-m5D

(g) MAVEN-r2D (h) MAVEN-r3D (i) MAVEN-r5D

Figure 4.14: Visual comparison of image samples from the CXR dataset against those generated by the different models.

4.3.3.3 CXR

With the CXR dataset, we used 522 labeled images and 4,694 unlabeled images. All the models were trained for 150 epochs and then evaluated. We generated an equal number of new images as the training set size. Figure 4.14 presents a visual comparison of a random selection of generated and real images. The FID and DDD measurements were performed for the distributions of generated and real training samples, indicating that more realistic images were generated by the MAVEN models than by the GAN and VAE-GAN models. The FID and DDD scores presented in Table 4.6 show that the mean MAVEN-m3D model has the smallest FID and DDD scores.

The classification performance reported in Table 4.9 suggests that our MAVEN model-based classifiers are more accurate than the baseline GAN and VAE-GAN classifiers. Among all the models, the MAVEN-m3D classifier was the most accurate.

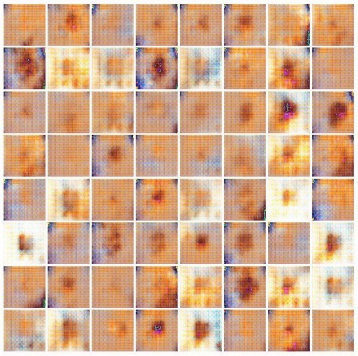
4.3.3.4 SLC

For the SLC dataset, we used 160 labeled images and 1,440 unlabeled images. All the models were trained for 150 epochs and then evaluated. We generated new images equal in number to the training set size. Figure 4.15 presents a visual comparison of randomly selected generated and real image samples.

The FID and DDD measurements for the distributions of generated and real training samples indicate that more realistic images were generated by the MAVEN models than by the GAN and VAE-GAN models. The FID and DDD scores presented in Table 4.6 show that the mean MAVEN-m3D model has the smallest FID and DDD scores.

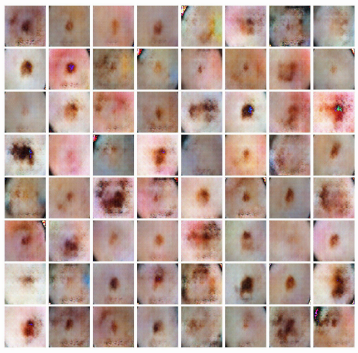
The classification performance reported in Table 4.10 suggests that our MAVEN model-based classifiers are more accurate than the baseline GAN and VAE-GAN classifiers. Among all the models, MAVEN-r3D is the most accurate in discriminating between non-melanoma and melanoma lesion images.

(a) Real samples



(b) DC-GAN

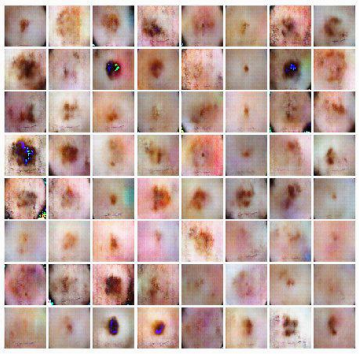
(c) VAE-GAN



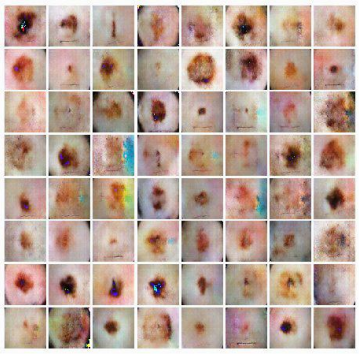
(d) MAVEN-m2D

(e) MAVEN-m3D

(f) MAVEN-m5D



(g) MAVEN-r2D



(h) MAVEN-r3D

(i) MAVEN-r5D

Figure 4.15: Visual comparison of image samples from the SLC dataset against those generated by the different models.

Table 4.10: Average cross-validation accuracy and class-wise F1 scores for the semi-supervised classification performance comparison of the DC-GAN, VAE-GAN, and MAVEN models using the SLC dataset.

Model	Accuracy	F1 scores	
		Non-melanoma	Melanoma
DC-GAN	0.802	0.890	0.120
VAE-GAN	0.810	0.890	0.012
MAVEN-m2D	0.815	0.900	0.016
MAVEN-m3D	0.814	0.900	0.110
MAVEN-m5D	0.812	0.900	0.140
MAVEN-r2D	0.808	0.890	0.260
MAVEN-r3D	**0.821**	0.900	0.020
MAVEN-r5D	0.797	0.890	0.040

4.4 Domain Generalization Without Domain-Specific Data

The progressive adversarial semantic segmentation (PASS) model was validated on two applications tasks: vascular segmentation and pulmonary segmentation. For the blood vessel segmentation, five different datasets were used (ARIA, CHASE, DRIVE, HRF, and STARE). For evaluation on the presence of domain shift, models trained on the ARIA and CHASE datasets, were tested against all five. Three chest X-Ray datasets (MCU, JSRT-V2, and CHN-V1) were used for the segmentation of lungs. Each of the chest X-ray datasets was used for separately training the models and testing was done against all three each time.

4.4.1 Implementation Details

Baselines: We employed a number of baseline and state-of-the-art models for medical image segmentation and domain adaptation: U-Net, U-Net with CRF, Pyramid U-Net (PU-Net), Progressive U-Net (ProgU-Net), Attention U-Net (AttnU-Net), Progressive U-Net with Side-Supervision (ProgU-NetSS), Adversarial U-Net (AU-Net), V-GAN, Adversarial Pyramid Progressive U-Net (APPU-Net), Unsupervised Domain Adaptation (UDA), ErrorNet, and Conditional Domain Adaptation with GAN (CoDAGAN).

Table 4.11: Comparison between PASS and other performance baselines for retinal vessel segmentation (Dice score).

Model	Train on →	CHASE						ARIA					
	Test on →	CHASE	DRIVE	ARIA	STARE	HRF	Avg.	CHASE	DRIVE	ARIA	STARE	HRF	Avg.
U-Net (Ronneberger et al., 2015b)		80.40	63.20	64.50	66.76	63.82	67.74	76.70	77.30	72.00	71.28	72.30	73.90
U-Net+CRF (Tajbakhsh et al., 2019b)		81.20	65.40	62.60	56.40	63.60	65.80	78.40	69.50	73.00	64.60	73.50	71.80
PU-Net (Fu et al., 2018)		81.58	64.04	63.03	66.20	62.66	67.50	76.70	77.30	72.00	71.28	72.30	73.90
AttnU-Net (Oktay et al., 2018)		81.37	65.23	62.91	64.28	65.72	67.90	76.70	77.30	72.00	71.28	72.30	73.90
ProgU-Net (Imran et al., 2019c)		82.91	61.02	63.28	66.58	63.43	67.44	47.21	64.54	70.56	66.57	60.17	61.81
ProgU-NetSS (Imran and Terzopoulos, 2019b)		80.16	62.13	62.41	65.44	63.78	66.78	57.96	65.93	74.47	69.08	60.08	65.50
V-GAN (Son et al., 2017)		79.70	71.50	64.20	61.00	66.40	68.50	68.70	75.80	69.90	66.20	69.30	70.00
AU-Net (Izadi et al., 2018)		82.20	63.20	61.84	67.17	63.37	67.56	68.06	70.21	78.12	74.95	69.69	72.21
APPU-Net (Imran and Terzopoulos, 2019b)		82.58	62.50	61.22	66.17	62.60	67.01	66.20	69.68	78.48	76.34	69.31	72.00
UDA (Dong et al., 2018)		72.30	69.30	68.20	64.70	67.40	68.40	71.50	72.90	73.20	71.30	70.70	71.90
ErrorNet (Tajbakhsh et al., 2019b)		81.50	73.20	66.50	65.20	68.60	71.00	76.70	78.90	72.00	74.00	72.60	74.80
PASS without $g(x)$		89.06	80.76	80.72	82.72	75.26	81.70	85.06	88.14	91.92	90.78	82.62	87.70
PASS		**91.96**	**84.96**	**84.18**	**86.84**	**78.57**	**85.30**	**86.32**	**90.55**	**92.08**	**91.50**	**83.15**	**88.72**

Inputs: All the images were resized and normalized to $256 \times 256 \times 3$ for the retinal images and $256 \times 256 \times 1$ for the chest X-rays before feeding them to the network.

Hyperparameters: We used the Adam optimizer with adaptive learning rate starting with initial rates of 0.01 for S, and 0.001 for D and E. The learning rates were decreased 90% after every 5 epochs with exponential decay. We apply the dropout with a rate of 0.25. We used $\lambda = 0.3$ for weighting $\mathcal{L}_{\mathcal{A}}$ in (3.18).

Machine Configuration: We implemented PASS in Tensorflow on a Nvidia Titan V GPU and a 64-bit Intel(R) Core(TM) i7-9700K CPU.

Evaluation: Along with qualitative visualization of segmentation masks and edges overlaid on the original input images, we calculated Dice score (DS), structural similarity index (SSIM), and average Hausdorff distance (HD) for the evaluation of pulmonary and vascular segmentations by different models.

4.4.2 Results

As reported in Tables 4.11 and 4.12, PASS outperforms all the baselines and state-of-the-art models for both retinal vessel and lung segmentation tasks.

In vessel segmentation, PASS achieved an overall average Dice score of 85.30 and cross-domain score of 83.74 (domain gap of 8.32), when trained on the small CHASE dataset.

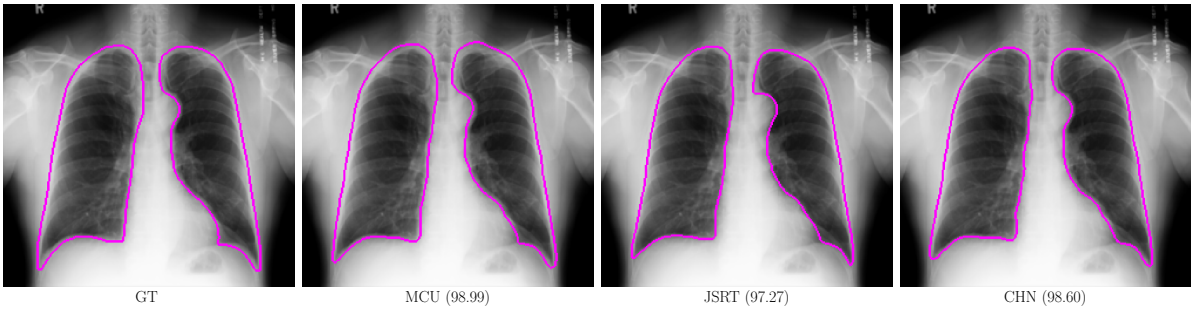


Figure 4.16: Consistency in the in-domain and cross-domain segmentation predictions by our PASS model: Test result for a chest X-ray from the MCU dataset when the model is trained on the MCU, JSRT, and CHN datasets.

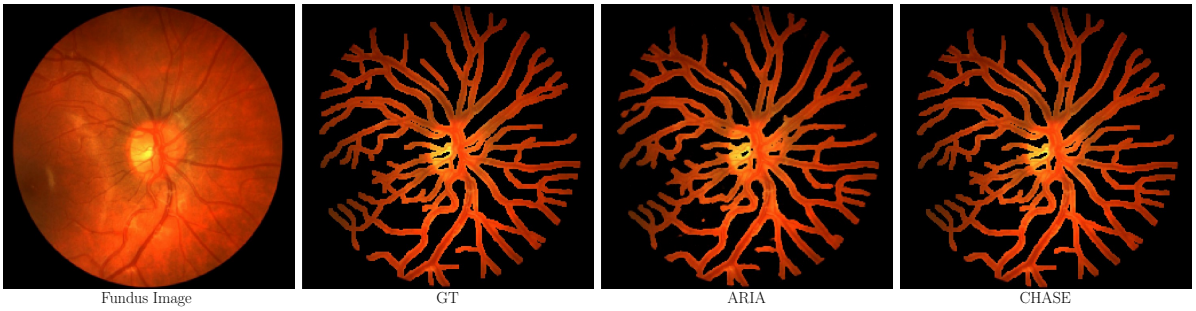


Figure 4.17: Visualization of retinal vessel segmentation by PASS from a fundus image when trained on the ARIA and CHASE datasets.

Table 4.12: Comparison between PASS and other performance baselines for lung segmentation from chest X-ray images (Dice score).

Model	Train on →	MCU				JSRT				CHN			
	Test on →	MCU	JSRT	CHN	Avg.	MCU	JSRT	CHN	Avg.	MCU	JSRT	CHN	Avg.
U-Net (Ronneberger et al., 2015b)		97.67	39.39	94.48	77.18	92.00	95.02	90.54	92.58	93.72	43.46	95.84	77.67
PU-Net (Fu et al., 2018)		97.89	21.24	**97.84**	72.33	84.97	94.94	73.68	84.53	93.57	73.88	95.90	87.78
AttnU-Net (Oktay et al., 2018)		97.86	30.31	94.07	74.08	6.70	94.95	65.00	55.55	81.25	74.24	95.56	83.68
ProgU-Net (Imran et al., 2019c)		97.83	10.98	91.32	66.71	34.89	95.20	86.28	72.12	84.79	60.03	95.35	80.06
ProgU-NetSS (Imran and Terzopoulos, 2019b)		97.90	33.98	95.32	75.33	13.16	95.09	65.00	57.75	94.24	67.29	95.63	85.72
AU-Net (Izadi et al., 2018)		97.86	94.68	95.08	95.87	89.12	97.85	92.46	93.14	95.58	95.88	96.22	95.89
APPU-Net (Imran and Terzopoulos, 2019b)		97.81	95.07	94.77	95.88	90.46	97.80	91.76	93.34	95.72	96.25	96.11	96.03
CyUDA (Chen et al., 2018)		95.61	92.84	–	94.23	–	–	–	–	–	–	–	–
SeUDA (Chen et al., 2018)		95.61	94.51	–	–	–	–	–	–	–	–	–	–
CoDAGAN (Oliveira et al., 2019)		–	–	–	–	84.58	96.45	88.99	90.01	–	–	–	–
PASS without $g(x)$		97.74	96.43	96.76	96.98	95.11	98.26	95.92	96.43	96.62	96.11	97.61	96.68
PASS		**98.22**	**96.56**	97.24	**97.34**	**95.70**	**98.27**	**96.06**	**96.68**	**97.27**	**97.15**	**97.65**	**97.36**

Table 4.13: Comparison between PASS and other performance baselines for pulmonary segmentation (HD score).

Model	Train on →	MCU				JSRT				CHN			
	Test on →	MCU	JSRT	CHN	Avg.	MCU	JSRT	CHN	Avg.	MCU	JSRT	CHN	Avg.
U-Net		3.689	9.948	4.849	6.162	7.923	4.128	5.594	5.882	4.874	9.161	4.378	6.138
PU-Net		3.723	11.049	5.346	6.706	10.926	4.159	6.618	7.234	5.347	7.438	4.362	5.716
AttnU-Net		3.841	10.189	5.057	6.362	11.628	4.181	7.374	7.728	6.989	7.257	4.453	6.233
ProgU-Net		3.729	11.706	5.564	6.999	10.491	4.052	5.700	6.748	6.788	8.702	4.494	6.661
ProgU-NetSS		3.874	9.907	4.673	6.151	11.602	4.117	7.509	7.743	4.885	7.906	4.362	5.718
AU-Net		3.780	4.625	4.651	4.352	4.879	3.705	5.126	4.570	4.327	4.499	4.301	4.376
APPU-Net		3.736	4.537	4.725	4.333	4.732	3.706	5.055	4.498	4.485	**4.278**	4.401	4.388
PASS without $g(x)$		3.332	4.552	4.857	4.247	5.109	3.867	5.138	4.705	4.580	4.441	4.545	4.522
PASS		**3.262**	**3.996**	**4.675**	**3.978**	**4.680**	**3.702**	**5.021**	**4.468**	**4.269**	4.397	**4.112**	**4.259**

Table 4.14: Comparison between PASS and other performance baselines for pulmonary segmentation (SSIM score).

Model	Train on →	MCU				JSRT				CHN			
	Test on →	MCU	JSRT	CHN	Avg.	MCU	JSRT	CHN	Avg.	MCU	JSRT	CHN	Avg.
U-Net		**0.968**	0.754	0.935	0.886	0.803	0.949	0.908	0.887	0.929	0.738	0.949	0.872
PU-Net		0.967	0.672	0.921	0.853	0.698	0.948	0.839	0.828	0.931	0.836	0.949	0.905
AttnU-Net		0.967	0.699	0.926	0.864	0.679	0.948	0.820	0.816	0.869	0.837	0.946	0.884
ProgU-Net		**0.968**	0.656	0.919	0.848	0.732	0.951	0.889	0.857	0.886	0.837	0.946	0.889
ProgU-NetSS		**0.968**	0.716	0.941	0.875	0.689	0.949	0.828	0.822	0.934	0.822	0.948	0.901
AU-Net		0.967	0.9191	0.936	0.941	0.892	**0.961**	0.916	0.923	0.942	0.937	0.951	0.943
APPU-Net		0.967	0.922	0.934	0.941	0.902	0.960	0.913	0.925	**0.944**	**0.940**	0.950	0.945
PASS without $g(x)$		0.962	0.921	0.932	0.938	0.913	0.953	**0.921**	0.929	0.939	0.931	0.945	0.938
PASS		**0.968**	**0.923**	**0.942**	**0.944**	**0.918**	0.955	0.920	**0.931**	0.942	0.939	**0.956**	**0.946**

Table 4.15: Comparison between PASS and other performance baselines for retinal vessel segmentation (SSIM score).

Model	Train on →	CHASE						ARIA					
	Test on →	CHASE	DRIVE	ARIA	STARE	HRF	Avg.	CHASE	DRIVE	ARIA	STARE	HRF	Avg.
U-Net		0.812	0.645	**0.665**	0.684	0.507	0.663	0.682	0.734	0.791	0.776	0.571	0.711
PU-Net		0.815	0.678	0.666	0.699	0.501	0.672	0.679	0.722	0.769	0.751	0.537	0.692
AttnU-Net		0.808	0.645	0.648	0.687	0.498	0.657	0.675	0.725	0.764	0.753	0.543	0.692
ProgU-Net		0.819	0.611	0.653	0.679	0.490	0.650	0.677	0.739	0.791	0.774	0.595	0.715
ProgU-NetSS		0.802	0.624	0.652	0.659	0.501	0.648	0.698	0.741	0.797	0.783	0.586	0.721
AU-Net		0.812	0.624	0.641	0.658	0.482	0.643	0.706	0.738	0.789	0.779	**0.632**	0.729
APPU-Net		0.820	0.619	0.636	0.654	0.484	0.643	0.695	0.738	0.796	0.789	**0.632**	0.730
PASS without $g(x)$		0.827	0.682	0.661	0.699	0.505	0.675	0.696	0.742	**0.798**	0.791	0.610	0.725
PASS		**0.830**	**0.685**	0.660	**0.701**	**0.510**	**0.677**	**0.707**	**0.772**	0.797	**0.802**	0.611	**0.738**

Table 4.16: Comparison between PASS and other performance baselines for retinal vessel segmentation (average HD score).

Model	Train on →	CHASE						ARIA					
	Test on →	CHASE	DRIVE	ARIA	STARE	HRF	Avg.	CHASE	DRIVE	ARIA	STARE	HRF	Avg.
U-Net		7.929	8.306	7.490	7.807	8.916	8.089	9.177	8.095	6.404	7.209	9.017	7.980
PU-Net		7.287	8.260	7.526	7.551	8.815	7.887	9.381	8.247	6.905	7.420	9.403	8.271
AttnU-Net		7.436	8.515	7.672	7.899	8.982	8.100	9.765	8.297	7.186	7.691	9.543	8.496
ProgU-Net		7.233	8.408	7.613	7.850	9.000	8.021	9.400	8.144	6.627	7.316	9.133	8.124
ProgU-NetSS		7.498	8.478	7.560	8.055	9.012	8.121	8.857	8.072	6.478	7.089	9.143	7.928
AU-Net		7.406	8.482	7.665	7.792	8.946	8.058	8.508	7.770	6.448	6.606	8.510	7.568
APPU-Net		7.196	8.492	7.732	7.824	8.979	8.045	8.486	8.009	6.344	6.519	8.655	7.602
PASS without $g(x)$		7.154	7.809	7.397	**7.346**	8.261	7.593	8.349	8.132	**6.301**	**6.120**	8.074	7.396
PASS		**7.129**	**7.341**	**7.187**	7.745	**8.028**	**7.486**	**8.285**	**8.012**	6.336	6.123	**7.399**	**7.231**

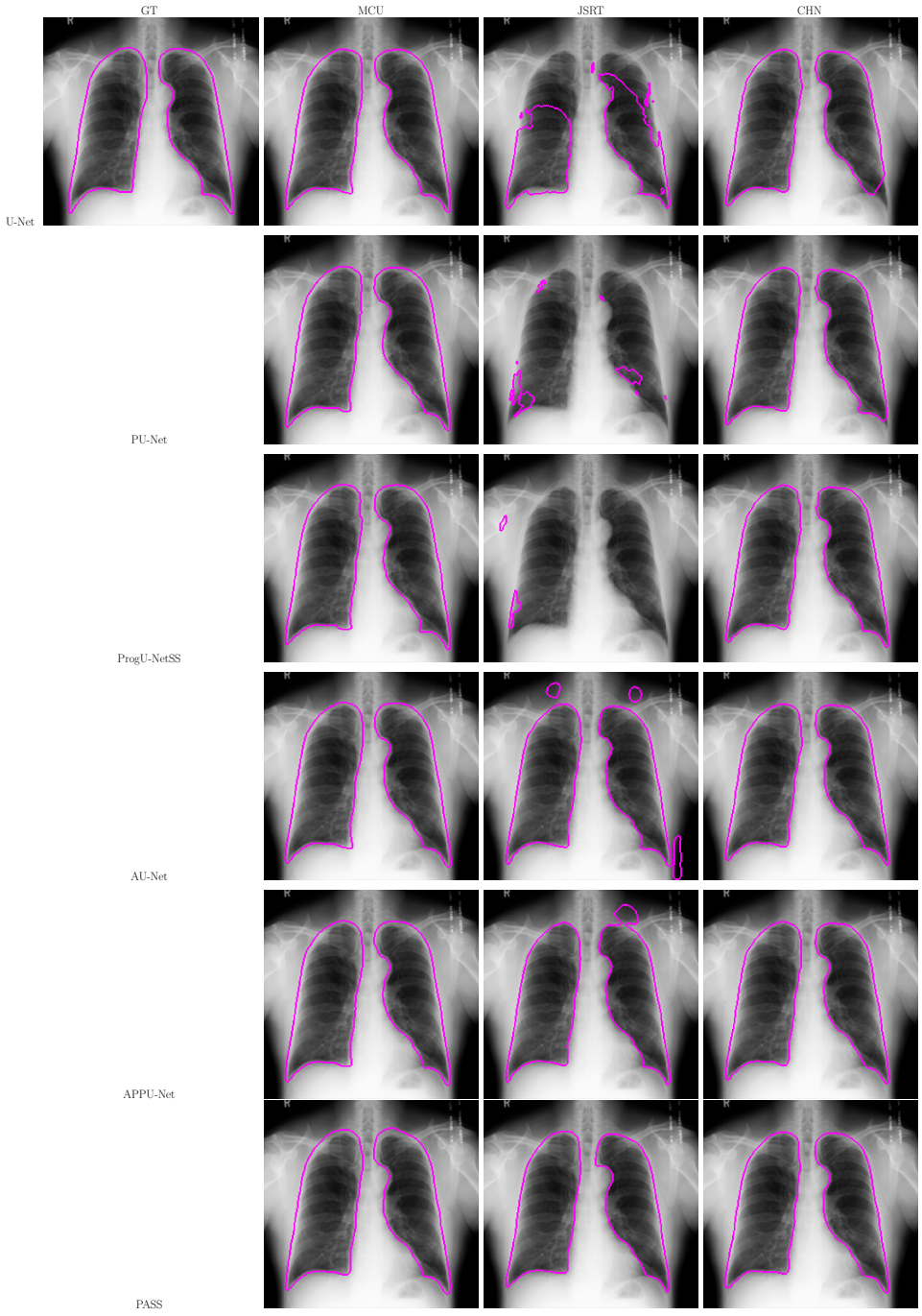


Figure 4.18: Visualization of lung boundaries for the predicted segmentations of a chest X-Ray image from the MCU dataset when trained on the MCU, JSRT, and CHN datasets.

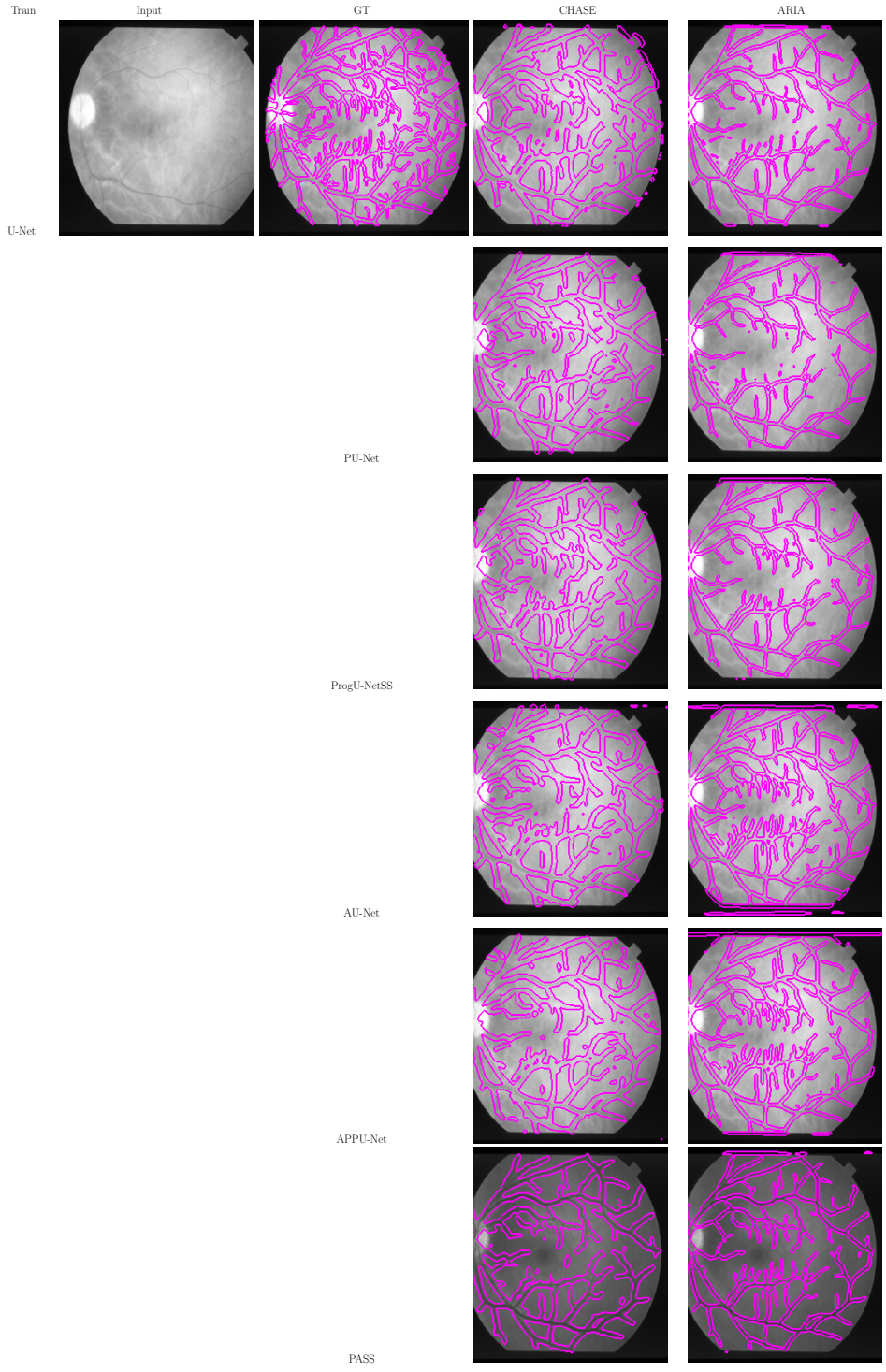


Figure 4.19: Visualization of vascular segmentation from fundoscopic image from the STARE dataset when the models are trained on the CHASE and ARIA datasets.

When trained on the relatively larger ARIA dataset, the overall average Dice score of 88.72 and cross-domain score of 87.88 are achieved with domain gap of 4.22. This demonstrates the effectiveness of PASS across the domains as it is capable of performing the semantic segmentation with the ability to learn diversity in shapes and distributions. Similarly, in the segmentation of lungs, PASS outperformed all the baselines and state-of-the-art models (Table 4.12).

Along with Figs. 4.16 and 4.17, Figure 4.18 (edges) shows the consistency of PASS in segmenting lungs from the chest X-ray and Figure 4.19 (edges) shows the consistency of PASS in segmenting retinal vessels from both in-domain and cross-domains irrespective of the varying imaging configurations and abnormalities. While some of the baseline models completely failed in evaluations on the JSRT dataset when trained on MCU and vice-versa, PASS consistently performs better in either scenario. With PASS, we have a domain gap of only 1.32 when trained on MCU, 2.39 when trained on JSRT, and 0.44 when trained on CHN. The poorer performance of the PASS model without $g(x)$ justifies the inclusion of this transformation function and the logit-wise distribution matching.

The HD and SSIM scores are reported for lung segmentation in Table 4.16 and Table 4.14, and for retinal vessel segmentation in Table 4.16 and Table 4.15. The consistently lower HD scores and higher SSIM scores compared to the baseline and state-of-the-art models provide further evidence of the superior performance by our PASS model.

4.5 Semi-Supervised Multi-Task Learning

Three publicly available chest X-ray datasets (MCU, CHN-V2, and JSRT-V1), and the combined Chest datasets were used for evaluating semi-supervised multi-task learning in jointly segmenting lungs and classification of lung abnormalities.

Figure 4.20: Visual comparison of the lung segmentation by the proposed APPAU-Net model with TV loss and KLTV loss. The predicted lung mask with TV and KLTV losses are overlaid with the ground truth mask.

4.5.1 Implementation Details

For the supervised segmentation, we used the proposed PPAU-Net model and KLTV as a loss function. We compared against all the preliminary segmentation models and losses. Then we performed semi-supervised multi-tasking for semi-supervised diease classification and lung segmentation from chest X-ray images. Except the Chest dataset, all datasets were used for binary classification (normal, tuberculosis/nodule), while the Chest dataset was used for 3-class classification (normal, nodule, tuberculosis). The X-ray images were normalized and resized to $128 \times 128 \times 1$ pixels. We consistently used the Adam optimizer with momentum of 0.9 and learning rates of $1e-5$ (segmentor), and $1e-4$ (discriminator) for multi-tasking. Each model was trained using a minibatch size of 16. All the convolutional layers were followed by batch-normalization except the convolutions for generating side-outputs. We performed droptout with a rate of 0.4 in the discriminator. Each model was evaluated using the best model selected based on validation performance, after running for 300 epochs. For the classification task, along with overall accuracy, we also reported class-wise F1 scores. For the segmentation evaluation, we used a number of performance metrics: Dice similarity (DS), Average Hausdorff distance (HD), Jaccard score (JS), Sensitivity (SN), Specificity (SP), F1 score, Structural Similarity Measure (SSIM), Precision (PR), and Recall (RE) scores.

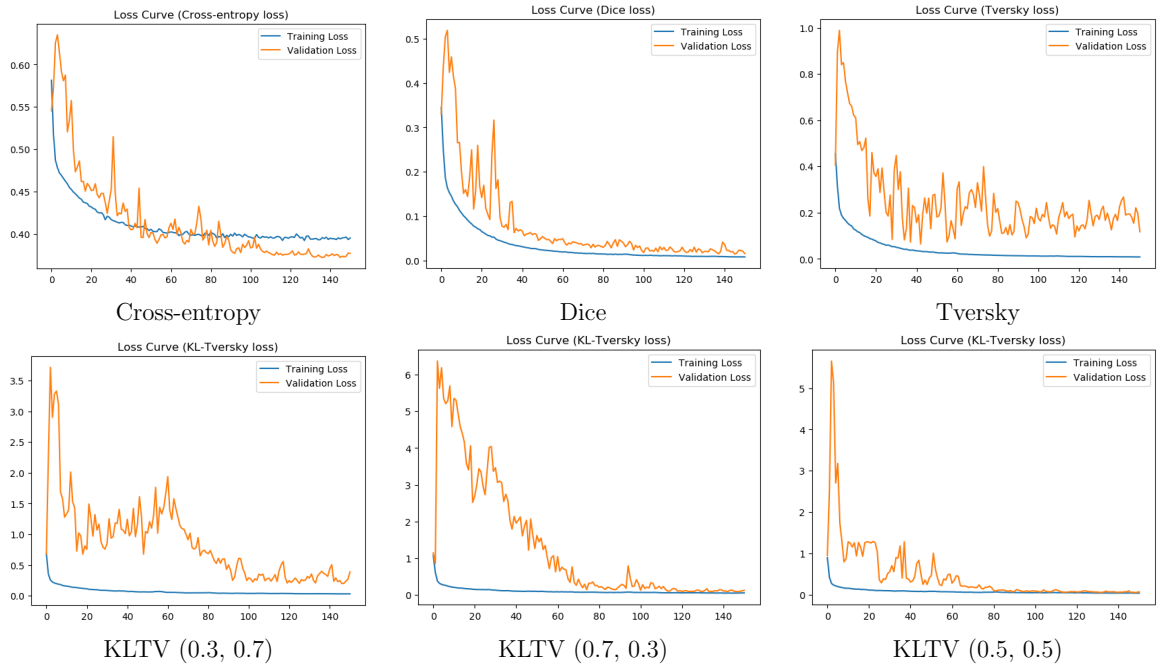


Figure 4.21: Training and Validation loss plots for the U-Net model with varying loss functions using the MCU dataset.

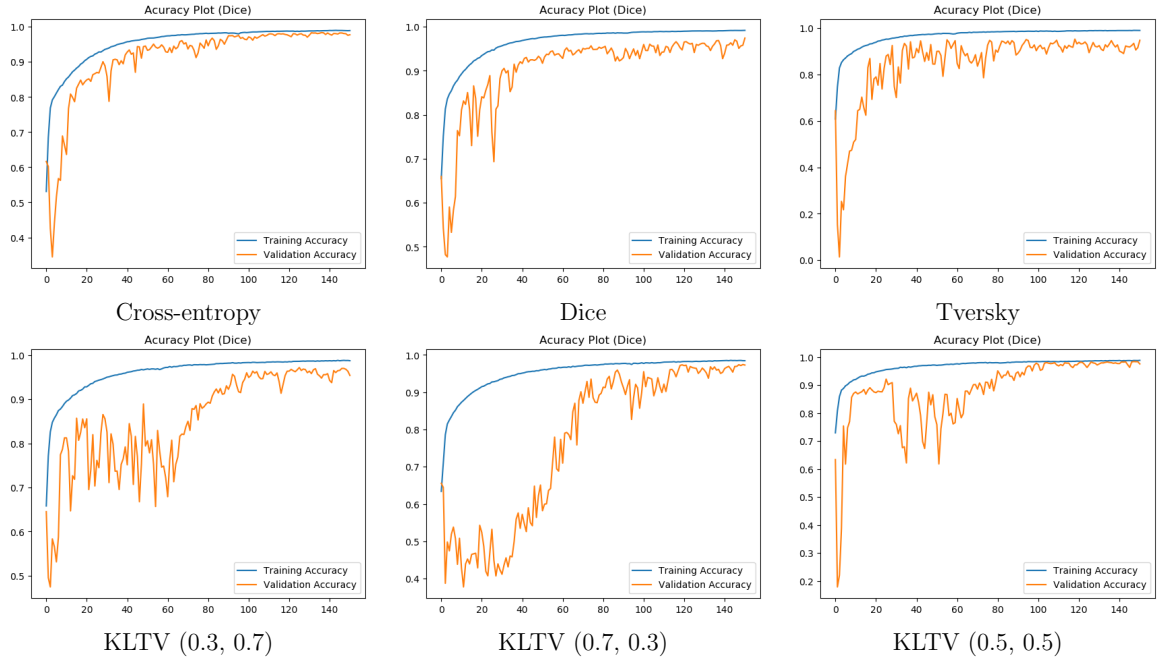


Figure 4.22: Training and Validation accuracy plots for the U-Net model with varying loss functions using the MCU dataset.

Table 4.17: Segmentation-Only performance evaluations for different models with varying loss functions on the MCU dataset.

Model	DS	JS	SSIM	F1	HD	SN	SP	PR	RC
U-Net-XE	0.991	0.983	0.951	0.967	2.896	0.956	0.992	0.977	0.957
U-Net-Dice	0.991	0.983	0.952	0.967	2.932	0.968	0.988	0.966	0.970
U-Net-TV	0.991	0.983	0.950	0.966	2.968	0.965	0.989	0.968	0.965
U-Net-KLTV	0.990	0.980	0.947	0.962	3.009	0.966	0.985	0.958	0.966
Pyramid U-Net-XE	0.991	0.983	0.951	0.966	2.837	0.956	0.992	0.977	0.956
Pyramid U-Net-Dice	0.992	0.985	0.956	0.970	2.781	0.968	0.990	0.972	0.968
Pyramid U-Net-TV	0.990	0.979	0.945	0.960	2.814	0.968	0.983	0.952	0.968
Pyramid U-Net-KLTV	0.991	0.983	0.951	0.966	2.858	0.951	0.994	0.983	0.951
Progressive U-Net-XE	0.991	0.983	0.950	0.966	2.888	0.966	0.988	0.966	0.966
Progressive U-Net-Dice	0.991	0.982	0.950	0.965	3.019	0.955	0.991	0.974	0.955
Progressive U-Net-TV	0.991	0.982	0.949	0.966	2.856	0.972	0.986	0.959	0.972
Progressive U-Net-KLTV	0.987	0.974	0.935	0.949	2.973	0.953	0.981	0.944	0.953
Attention U-Net-XE	0.990	0.981	0.946	0.963	3.034	0.957	0.990	0.969	0.967
Attention U-Net-Dice	0.988	0.977	0.938	0.956	3.316	0.946	0.988	0.965	0.946
Attention U-Net-TV	0.984	0.968	0.922	0.937	3.768	0.915	0.987	0.960	0.915
Attention U-Net-KLTV	0.990	0.980	0.941	0.960	3.063	0.957	0.987	0.962	0.957
Pyramid Progressive U-Net-XE	0.991	0.982	0.949	0.965	2.847	0.967	0.987	0.963	0.967
Pyramid Progressive U-Net-Dice	0.988	0.976	0.933	0.954	3.239	0.957	0.983	0.950	0.957
Pyramid Progressive U-Net-TV	0.984	0.968	0.919	0.938	3.978	0.932	0.981	0.944	0.932
Pyramid Progressive U-Net-KLTV	0.987	0.974	0.930	0.948	3.438	0.934	0.988	0.963	0.934
Pyramid Attention U-Net-XE	0.988	0.976	0.935	0.955	2.922	0.965	0.980	0.945	0.965
Pyramid Attention U-Net-Dice	0.989	0.978	0.941	0.957	3.367	0.948	0.989	0.967	0.948
Pyramid Attention U-Net-TV	0.991	0.982	0.946	0.965	2.969	0.971	0.985	0.958	0.971
Pyramid Attention U-Net-KLTV	0.988	0.976	0.934	0.952	3.313	0.945	0.986	0.960	0.945
Progressive Attention U-Net-XE	0.986	0.973	0.927	0.947	3.434	0.941	0.984	0.954	0.941
Progressive Attention U-Net-Dice	0.982	0.964	0.914	0.931	3.697	0.942	0.972	0.921	0.942
Progressive Attention U-Net-TV	0.987	0.973	0.927	0.949	3.397	0.957	0.979	0.941	0.957
Progressive Attention U-Net-KLTV	0.983	0.966	0.917	0.930	3.748	0.886	0.993	0.978	0.886
Pyramid Progressive Attention U-Net-XE	0.984	0.968	0.918	0.937	3.576	0.927	0.982	0.947	0.927
Pyramid Progressive Attention U-Net-Dice	0.988	0.977	0.936	0.955	3.203	0.952	0.985	0.957	0.952
Pyramid Progressive Attention U-Net-TV	0.989	0.978	0.936	0.958	3.143	0.967	0.982	0.949	0.967
Pyramid Progressive Attention U-Net-KLTV	0.984	0.970	0.921	0.939	3.470	0.915	0.988	0.964	0.915

Table 4.18: Segmentation-Only performance evaluations for different models with varying loss functions on the CHN dataset.

Model	DS	JS	SSIM	F1	HD	SN	SP	PR	RC
U-Net-XE	0.969	0.940	0.878	0.960	3.706	0.964	0.867	0.956	0.964
U-Net-Dice	0.966	0.935	0.871	0.957	3.959	0.973	0.820	0.942	0.973
U-Net-TV	0.964	0.931	0.860	0.955	4.181	0.975	0.799	0.936	0.975
U-Net-KLTV	0.960	0.923	0.850	0.950	4.023	0.963	0.803	0.936	0.963
Pyramid U-Net-XE	0.966	0.934	0.869	0.957	3.932	0.966	0.838	0.947	0.966
Pyramid U-Net-Dice	0.966	0.933	0.868	0.956	4.101	0.972	0.818	0.941	0.972
Pyramid U-Net-TV	0.964	0.930	0.863	0.954	4.121	0.978	0.787	0.932	0.978
Pyramid U-Net-KLTV	0.967	0.936	0.869	0.957	3.682	0.956	0.875	0.958	0.956
Progressive U-Net-XE	0.965	0.933	0.868	0.956	3.951	0.968	0.828	0.944	0.968
Progressive U-Net-Dice	0.968	0.939	0.873	0.959	3.776	0.964	0.864	0.955	0.964
Progressive U-Net-TV	0.955	0.913	0.842	0.945	4.408	0.984	0.701	0.908	0.984
Progressive U-Net-KLTV	0.964	0.931	0.860	0.954	3.758	0.951	0.873	0.958	0.951
Attention U-Net-XE	0.956	0.915	0.841	0.945	4.447	0.972	0.742	0.919	0.972
Attention U-Net-Dice	0.966	0.935	0.866	0.957	3.978	0.969	0.831	0.945	0.969
Attention U-Net-TV	0.958	0.919	0.842	0.948	4.562	0.983	0.725	0.915	0.983
Attention U-Net-KLTV	0.965	0.933	0.862	0.955	3.684	0.946	0.894	0.964	0.946
Pyramid Progressive U-Net-XE	0.967	0.937	0.871	0.958	3.836	0.967	0.846	0.950	0.967
Pyramid Progressive U-Net-Dice	0.966	0.935	0.871	0.957	3.955	0.973	0.820	0.942	0.973
Pyramid Progressive U-Net-TV	0.961	0.925	0.854	0.951	4.311	0.979	0.763	0.926	0.979
Pyramid Progressive U-Net-KLTV	0.966	0.934	0.864	0.956	3.725	0.952	0.879	0.959	0.952
Pyramid Attention U-Net-XE	0.967	0.937	0.869	0.958	3.919	0.958	0.874	0.958	0.958
Pyramid Attention U-Net-Dice	0.963	0.928	0.856	0.953	4.296	0.966	0.816	0.941	0.966
Pyramid Attention U-Net-TV	0.951	0.906	0.829	0.941	4.892	0.988	0.661	0.898	0.988
Pyramid Attention U-Net-KLTV	0.966	0.937	0.863	0.955	3.664	0.946	0.896	0.965	0.990
Progressive Attention U-Net-XE	0.960	0.923	0.849	0.950	4.623	0.969	0.782	0.931	0.970
Progressive Attention U-Net-Dice	0.963	0.928	0.854	0.953	4.358	0.970	0.801	0.936	0.970
Progressive Attention U-Net-TV	0.950	0.906	0.828	0.940	4.772	0.987	0.660	0.897	0.987
Progressive Attention U-Net-KLTV	0.961	0.925	0.849	0.950	3.750	0.942	0.875	0.958	0.942
Pyramid Progressive Attention U-Net-XE	0.968	0.939	0.877	0.960	3.789	0.967	0.857	0.953	0.967
Pyramid Progressive Attention U-Net-Dice	0.968	0.937	0.873	0.959	3.887	0.967	0.849	0.951	0.967
Pyramid Progressive Attention U-Net-TV	0.954	0.913	0.838	0.944	4.523	0.983	0.700	0.908	0.983
Pyramid Progressive Attention U-Net-KLTV	0.968	0.936	0.870	0.957	3.801	0.959	0.865	0.955	0.960

Table 4.19: Segmentation-only performance evaluations for different models with varying loss functions on the JSRT dataset.

Model	DS	JS	SSIM	F1	HD	SN	SP	PR	RC
U-Net-XE	0.991	0.983	0.946	0.988	2.678	0.988	0.970	0.987	0.988
U-Net-Dice	0.990	0.981	0.942	0.986	2.730	0.991	0.959	0.982	0.999
U-Net-TV	0.989	0.979	0.937	0.985	2.804	0.989	0.956	0.981	0.989
U-Net-KLTV	0.990	0.980	0.940	0.986	2.553	0.977	0.988	0.995	0.977
Pyramid U-Net-XE	0.991	0.983	0.947	0.988	2.670	0.989	0.969	0.986	0.990
Pyramid U-Net-Dice	0.990	0.981	0.942	0.986	2.857	0.988	0.966	0.985	0.988
Pyramid U-Net-TV	0.990	0.981	0.942	0.986	2.742	0.995	0.950	0.978	0.994
Pyramid U-Net-KLTV	0.990	0.980	0.940	0.986	2.729	0.982	0.975	0.989	0.982
Progressive U-Net-XE	0.991	0.982	0.945	0.987	2.678	0.988	0.968	0.986	0.988
Progressive U-Net-Dice	0.990	0.981	0.941	0.986	2.644	0.987	0.968	0.986	0.987
Progressive U-Net-TV	0.989	0.978	0.934	0.984	2.911	0.993	0.943	0.975	0.993
Progressive U-Net-KLTV	0.989	0.977	0.933	0.984	2.756	0.988	0.953	0.979	0.988
Attention U-Net-XE	0.991	0.982	0.943	0.987	2.738	0.986	0.973	0.988	0.986
Attention U-Net-Dice	0.984	0.968	0.911	0.977	3.259	0.993	0.912	0.962	0.993
Attention U-Net-TV	0.988	0.977	0.929	0.983	2.882	0.993	0.940	0.974	0.993
Attention U-Net-KLTV	0.989	0.977	0.932	0.984	2.781	0.981	0.970	0.986	0.981
Pyramid Progressive U-Net-XE	0.990	0.981	0.942	0.986	2.692	0.985	0.971	0.987	0.985
Pyramid Progressive U-Net-Dice	0.991	0.982	0.945	0.987	2.730	0.990	0.966	0.985	0.990
Pyramid Progressive U-Net-TV	0.991	0.981	0.942	0.987	2.735	0.990	0.962	0.983	0.990
Pyramid Progressive U-Net-KLTV	0.990	0.980	0.939	0.985	2.652	0.980	0.980	0.991	0.980
Pyramid Attention U-Net-XE	0.991	0.982	0.942	0.987	2.797	0.989	0.964	0.984	0.989
Pyramid Attention U-Net-Dice	0.989	0.979	0.934	0.985	2.861	0.987	0.961	0.983	0.987
Pyramid Attention U-Net-TV	0.990	0.980	0.938	0.986	2.846	0.993	0.952	0.979	0.993
Pyramid Attention U-Net-KLTV	0.988	0.976	0.934	0.982	2.805	0.986	0.957	0.987	0.985
Progressive Attention U-Net-XE	0.990	0.981	0.940	0.986	2.838	0.981	0.985	0.987	0.985
Progressive Attention U-Net-Dice	0.988	0.975	0.928	0.982	3.102	0.992	0.940	0.973	0.992
Progressive Attention U-Net-TV	0.989	0.977	0.933	0.984	3.022	0.990	0.949	0.978	0.990
Progressive Attention U-Net-KLTV	0.990	0.981	0.940	0.986	2.595	0.980	0.983	0.992	0.800
Pyramid Progressive Attention U-Net-XE	0.991	0.982	0.944	0.987	2.706	0.988	0.970	0.987	0.988
Pyramid Progressive Attention U-Net-Dice	0.988	0.976	0.928	0.983	3.068	0.988	0.949	0.972	0.988
Pyramid Progressive Attention U-Net-TV	0.991	0.981	0.941	0.987	2.768	0.992	0.958	0.981	0.992
Pyramid Progressive Attention U-Net-KLTV	0.990	0.979	0.937	0.985	2.751	0.987	0.959	0.982	0.987

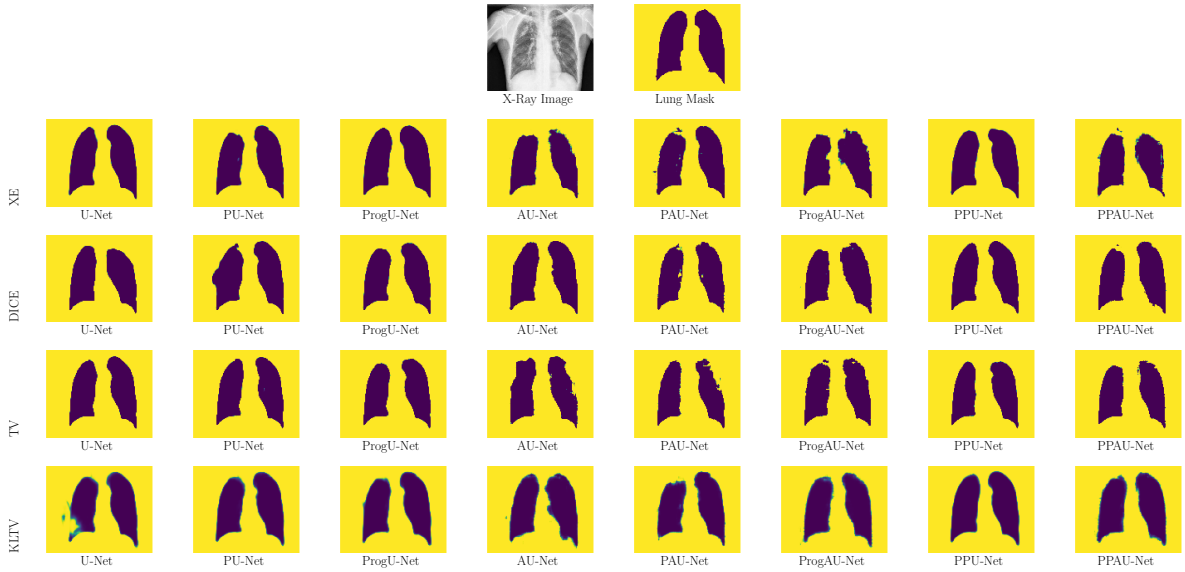


Figure 4.23: Visual comparison of the lung segmentation in an abnormal (TB) X-Ray image, for different models against the Chest dataset with varying loss functions: XE (cross-entropy loss), DICE (DICE loss), TV (Tversky loss), and KLTV (KL divergence-Tversky loss).

4.5.2 Segmentation-Only

At first, we evaluated the performance of our proposed PPAU-Net model for segmentation task only, and compared with the baseline models used incrementally. Tables 4.17-4.20 report the performance measures of different models for the segmentation-only task varying choices of loss (TV and KLTV). Figures 4.22-4.21 illustrate the accuracy and loss plots (training/validation) for varying choice of the weights between the absolute KL divergence and Tversky loss. As shown in the tables, our model is competitive with the other models in each setting.

4.5.3 Segmentation and Classification

For semi-supervised multi-tasking, we used our PPAU-Net model in an adversarial training mechanism. Along with TV loss, we used XETV loss (Zhu et al., 2019) and the proposed KLTV loss. Only 10% labeled data were used from the training set for every dataset. As the tabulated classification and segmentation results from Table 4.21 show, the APPAU-Net model with the new KLTV loss consistently outperformed APPAU-Net model with

Table 4.20: Segmentation-Only performance evaluations for different models with varying loss functions on the Chest dataset.

Model	DS	JS	SSIM	F1	HD	SN	SP	PR	RC
U-Net-XE	0.968	0.0.939	0.882	0.954	3.599	0.963	0.893	0.944	0.963
U-Net-Dice	0.976	0.954	0.902	0.965	3.527	0.968	0.928	0.962	0.968
U-Net-TV	0.978	0.958	0.907	0.968	3.322	0.974	0.928	0.962	0.974
U-Net-KLTV	0.969	0.939	0.874	0.953	3.502	0.946	0.926	0.960	0.946
Pyramid U-Net-XE	0.976	0.54	0.899	0.965	3.496	0.968	0.928	0.962	0.968
Pyramid U-Net-Dice	0.979	0.959	0.909	0.968	3.343	0.968	0.941	0.968	0.968
Pyramid U-Net-TV	0.975	0.952	0.897	0.964	3.53	0.978	0.904	0.950	0.977
Pyramid U-Net-KLTV	0.972	0.946	0.884	0.958	3.365	0.951	0.936	0.966	0.951
Progessive U-Net-XE	0.973	0.947	0.885	0.958	3.644	0.945	0.950	0.972	0.945
Progessive U-Net-Dice	0.966	0.934	0.877	0.950	3.715	0.969	0.868	0.932	0.969
Progessive U-Net-TV	0.969	0.940	0.881	0.955	3.687	0.980	0.866	0.932	0.980
Progessive U-Net-KLTV	0.962	0.927	0.863	0.945	3.518	0.951	0.9882	0.938	0.951
Attention U-Net-XE	0.966	0.935	0.865	0.951	3.936	0.959	0.890	0.942	0.959
Attention U-Net-Dice	0.976	0.954	0.898	0.965	3.513	0.968	0.927	0.961	0.968
Attention U-Net-TV	0.970	0.941	0.878	0.956	3.643	0.972	0.883	0.940	0.972
Attention U-Net-KLTV	0.971	0.943	0.877	0.956	3.481	0.944	0.941	0.968	0.944
Pyramid Progressive U-Net-XE	0.973	0.947	0.891	0.960	3.602	0.967	0.910	0.953	0.967
Pyramid Progressive U-Net-Dice	0.977	0.954	0.901	0.965	3.528	0.970	0.925	0.960	0.970
Pyramid Progressive U-Net-TV	0.977	0.955	0.901	0.966	3.511	0.980	0.908	0.952	0.980
Pyramid Progressive U-Net-KLTV	0.975	0.949	0.876	0.958	3.475	0.968	0.912	0.957	0.966
Pyramid Attention U-Net-XE	0.968	0.938	0.879	0.953	3.527	0.966	0.884	0.940	0.966
Pyramid Attention U-Net-Dice	0.972	0.946	0.882	0.959	3.693	0.969	0.904	0.950	0.969
Pyramid Attention U-Net-TV	0.972	0.946	0.885	0.959	3.743	0.982	0.878	0.938	0.982
Pyramid Attention U-Net-KLTV	0.959	0.921	0.851	0.940	3.749	0.947	0.871	0.932	0.947
Progressive Attention U-Net-XE	0.954	0.912	0.841	0.934	4.237	0.960	0.822	0.910	0.960
Progressive Attention U-Net-Dice	0.974	0.949	0.891	0.961	3.611	0.970	0.909	0.952	0.970
Progressive Attention U-Net-TV	0.969	0.940	0.875	0.955	3.723	0.975	0.876	0.936	0.975
Progressive Attention U-Net-KLTV	0.946	0.899	0.823	0.924	3.712	0.932	0.838	0.915	0.932
Pyramid Progressive Attention U-Net-XE	0.966	0.934	0.871	0.951	3.822	0.962	0.885	0.940	0.962
Pyramid Progressive Attention U-Net-Dice	0.976	0.953	0.896	0.964	3.648	0.971	0.918	0.957	0.971
Pyramid Progressive Attention U-Net-TV	0.969	0.940	0.875	0.955	3.807	0.978	0.870	0.934	0.978
Pyramid Progressive Attention U-Net-KLTV	0.967	0.936	0.868	0.951	3.472	0.939	0.932	0.963	0.939

TV and XETV losses for all four datasets in both overlap and distance measures. It also suggests that the model with KLTV loss is better generalized in multi-task learning. While both TV and XETV losses tend to lose some accuracy because of additional classification task, KLTV still achieves good accuracy, which is even comparable to fully supervised segmentation models in Tables 4.17-4.20 and LF-segnet (Mittal et al., 2018). Visualizations of the segmented lungs in different models (see Figure 4.20) also confirms the superior performance of our APPAU-Net with KLTV loss compared to the TV loss.

Table 4.21: Performance evaluation of the APPAU-Net model for semi-supervised multi-tasking in different data settings.

Dataset	Model	Classification			Segmentation								
		Acc	PR	RE	DS	JS	SSIM	F1	HD	SN	SP	PR	RC
MCU	APPAU-Net-TV	**0.571**	0.690	0.290	0.956	0.916	0.815	0.814	4.514	0.800	0.988	0.953	0.800
	APPAU-Net-XETV	0.514	0.620	0.280	0.929	0.868	0.788	0.778	4.554	0.903	0.856	0.684	0.903
	APPAU-Net-KLTV	0.543	0.680	0.200	**0.974**	0.950	0.880	0.898	**3.914**	0.857	0.944	0.944	0.857
JSRT	APPAU-Net-TV	**0.758**	0.000	0.860	0.972	0.945	0.864	0.963	3.755	0.996	0.831	0.929	0.996
	APPAU-Net-XETV	**0.758**	0.000	0.860	0.975	0.952	0.878	0.966	3.489	0.995	0.857	0.939	0.994
	APPAU-Net-KLTV	**0.758**	0.000	0.860	**0.976**	0.953	0.885	0.966	**3.351**	0.975	0.904	0.958	0.975
CHN	APPAU-Net-TV	0.477	0.580	0.300	0.883	0.790	0.713	0.877	6.601	0.999	0.162	0.782	0.992
	APPAU-Net-XETV	**0.553**	0.670	0.290	0.889	0.800	0.720	0.882	6.372	0.997	0.205	0.791	0.997
	APPAU-Net-KLTV	0.508	0.530	0.490	**0.921**	0.853	0.746	0.910	**4.368**	0.992	0.434	0.841	0.992
Chest	APPAU-Net-TV	**0.776**	0.800	0.780	0.874	0.777	0.682	0.845	5.375	0.936	0.576	0.770	0.959
	APPAU-Net-XETV	0.732	0.81	0.70	0.923	0.862	0.768	0.890	4.692	0.974	0.632	0.823	0.954
	APPAU-Net-KLTV	0.750	0.770	0.750	**0.926**	0.863	0.780	0.903	**4.669**	0.979	0.645	0.838	0.953

4.6 Self-Supervised Semi-Supervised Multi-Task Learning

Two datasets—Chest and Spine—were used for evaluating the self-supervised semi-supervised multi-task learning models. For each dataset, we trained the models on its training set (labeled and unlabeled data), used its validation set to determine the hyper-parameters and for model selection, and evaluated the trained models on its test set. The class and segmentation label distributions demonstrated in Figure 4.24, shows the for either datasets.

4.6.1 Implementation Details

Inputs: All the images were normalized and resized to $128 \times 128 \times 1$ before feeding them to the models.

Model Architecture: As the segmentation mask generator we used a U-Net like encoder-decoder network with skip connections (Ronneberger et al., 2015b), and as the class discriminator we used another convolutional network (Conv-Net) (Imran and Terzopoulos, 2019b). We implemented the S^4MTL algorithm in Tensorflow running on a Tesla P40 GPU and a 64-bit Intel(R) Xeon(R) 440G CPU.

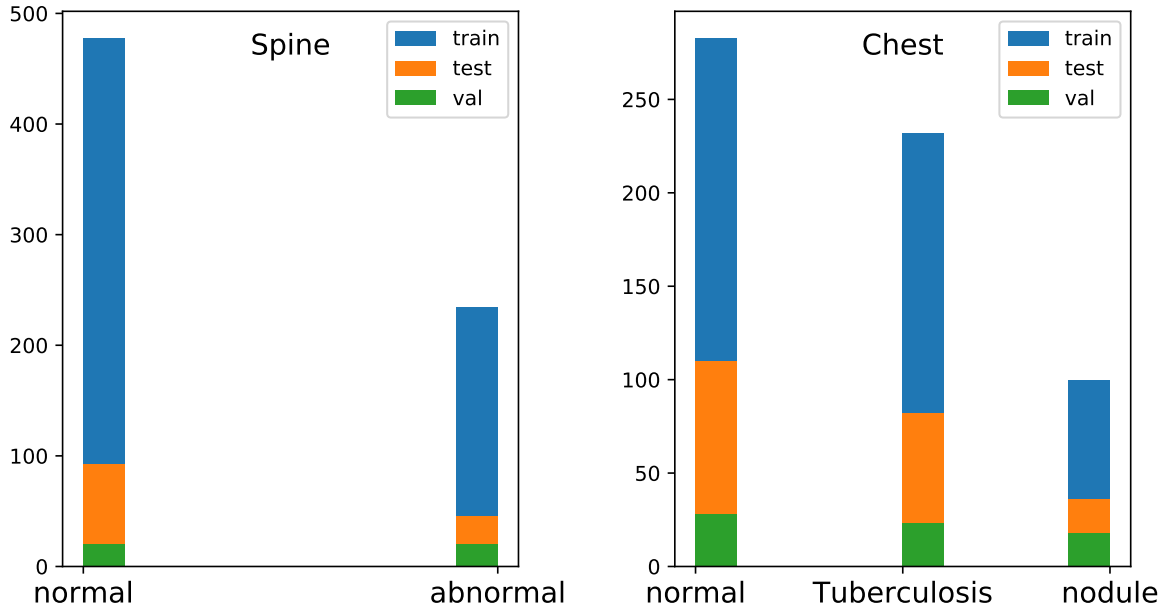


Figure 4.24: Class label distributions across different data splits for the Spine and Chest datasets.

Baselines: As baselines, we used the segmentation mask generator (U-Net) and class discriminator (Conv-Net) networks separately for single-task models both in supervised and partially-supervised manners. Using the same backbone network, we also trained a multitasking U-Net with a classification branch from its bottleneck layer (similarly as Y-Net (Mehta et al., 2018)), namely U-MTL. In addition, we experimented with another semi-supervised multitasking model without self-supervision and unsupervised segmentation losses. We call this the semi-supervised multitask learning (S^2MTL) model.

Training: Following our formulated problem, $\mathcal{D}_{\mathcal{L}}$ and $\mathcal{D}_{\mathcal{U}}$ are selected before training the models, rather than just masking some data out during training. We constrained all the semi-supervised models to having maximum 50% labeled data in order to hold $|\mathcal{D}_{\mathcal{L}}| \leq |\mathcal{D}_{\mathcal{U}}|$. The semi-supervised models (single-task or multitask) were trained on varying proportions of labeled data: 5%, 10%, 20%, 30%, and 50%. For example, when 10% is selected, 10% of the training data were used with their corresponding class and

segmentation labels, and 90% were used without any label information. We adapted the training signal annealing in our model using the logarithmic schedule Xie et al. (2019) with an adjustment for balancing between epochs and mini-batches. In our experiments, the signal annealing threshold $\eta_{(e,s)}$ is set to $1 - \exp\left(-\frac{s*e+1}{E*N}\right) * \left(1 - \frac{1}{n}\right) + \frac{1}{n}$, where s is the current step, e is the current epoch, E is total number of epochs, N is the training dataset size, and n is the number of classes. In training, the labeled data were selected such that every class has equal representation in the training data $\mathcal{D}_{\mathcal{L}}$. In the mask generator network, we used instance-normalization and ReLU activation. A dropout rate of 0.4 was applied after every convolutional layer.

Hyper-parameters: We used the Adam optimizer with adaptive learning rates for G and D. With initial learning rates $2e-3$ with momentum 0.9 for G and $1e-4$ with momentum 0.6 for D. The learning rates were adapted with exponential decay scheduled after every 2 epoch with decay rates of 0.9 and 0.5, respectively. Each model was trained with a mini-batch size of 16.

Evaluation: For classification, along with the overall accuracy, we recorded the class-wise F1 scores. Dice similarity (DS), average Hausdorff distance (HD), Jaccard index (JI), structural similarity measure (SSIM), precision (Prec), and recall (Rec) scores were used to evaluate the segmentation performances.

4.6.2 Classification Results

We validated our S^{4}MTL model on two separate datasets and the performance evaluations were compared against semi-supervised and fully-supervised models. Figure 4.27 shows the balanced optimization of two losses clearly depicting the generalization of the model for multitasking, even when the model was trained on 50% reduced labeled data. For both the Spine and Chest datasets, our S^{4}MTL model was found to be superior to all the baseline models (Figure 4.26). Table 4.22 compares the classification performances among all the models, and our model achieves better overall and class-wise accuracies for

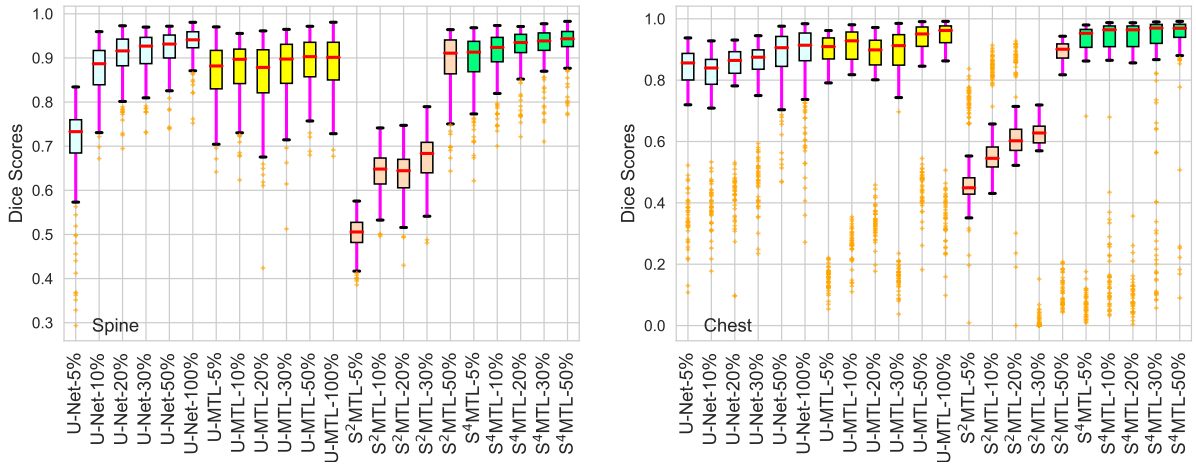


Figure 4.25: Consistent improvement in segmentation performance (Dice scores) by our S^{4}MTL model over baseline semi-supervised and fully-supervised (U-Net, U-MTL, S^{2}MTL) single/multi-tasks models with varying proportions of labeled training data.

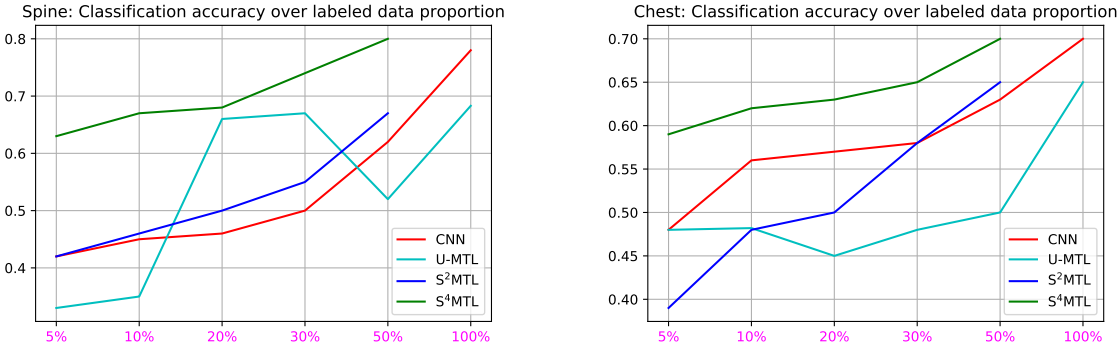


Figure 4.26: Consistent improvement in classification accuracy by our S^{4}MTL model over baseline semi-supervised and fully-supervised (CNN, U-MTL, S^{2}MTL) single/multi-task models with varying proportions of labeled training data.

both datasets, even better than the fully-supervised single-task model.

4.6.3 Segmentation Results

The segmentation performance was evaluated both qualitatively and quantitatively. As shown in Table 4.24 and 4.23, for all six performance measures, our S^{4}MTL model achieved the best scores compared to the semi-supervised, fully-supervised, single-task, and multitask models. Note that the Chest Dataset is the combination of three smaller datasets. The robustness of our S^{4}MTL model is confirmed by its consistent performance

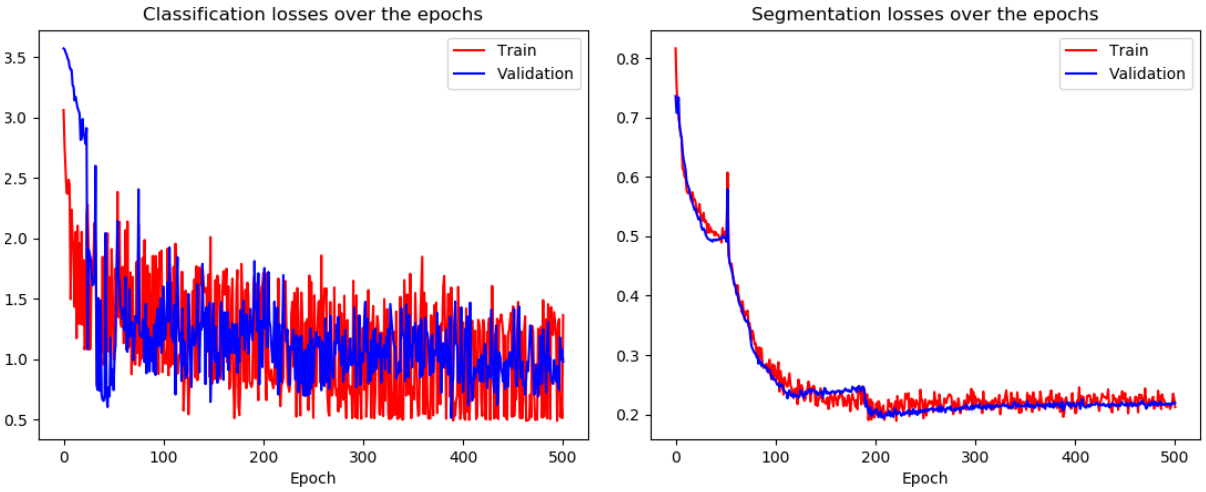


Figure 4.27: Satisfactory balance between training and validation losses for classification and segmentation in our S^{4}MTL model when it is trained only on 50% labeled data (both class and segmentation labels) from the training set

Table 4.22: Classification performance comparison of the S^{4}MTL model against the baseline models in different data settings with varying proportions of labeled data.

Type	Model	Spine			Chest			
		Accuracy	F1 (Normal)	F1 (Abnormal)	Accuracy	F1 (Normal)	F1 (TB)	F1 (Nodule)
Single-Task	Conv-Net-100%	0.780	0.740	0.800	**0.700**	0.530	0.720	0.840
	Conv-Net-50%	0.620	0.680	0.580	0.630	0.460	0.710	0.730
	Conv-Net-30%	0.500	0.000	0.630	0.580	0.590	0.360	0.710
	Conv-Net-20%	0.460	0.460	0.460	0.570	0.620	0.250	0.730
	Conv-Net-10%	0.450	0.490	0.410	0.560	0.630	0.180	0.710
	Conv-Net-5%	0.420	0.330	0.490	0.480	0.650	0.000	0.000
Multitask	UMTL-100%	0.683	0.810	0.008	0.650	0.610	0.290	0.780
	UMTL-50%	0.520	0.580	0.430	0.500	0.680	0.200	0.000
	UMTL-30%	0.670	0.800	0.000	0.480	0.650	0.000	0.000
	UMTL-20%	0.660	0.800	0.000	0.450	0.530	0.450	0.000
	UMTL-10%	0.350	0.030	0.051	0.482	0.650	0.000	0.000
	UMTL-5%	0.331	0.000	0.050	0.480	0.650	0.000	0.000
	S^{2}MTL-50%	0.670	0.800	0.000	0.650	0.610	0.290	0.780
	S^{2}MTL-30%	0.550	0.680	0.250	0.580	0.590	0.380	0.690
	S^{2}MTL-20%	0.500	0.000	0.630	0.500	0.570	0.410	0.490
	S^{2}MTL-10%	0.460	0.430	0.490	0.480	0.650	0.000	0.000
	S^{2}MTL-5%	0.420	0.330	0.500	0.390	0.470	0.350	0.280
	S^{4}MTL-50%	**0.800**	0.790	0.810	**0.700**	0.530	0.710	0.730
	S^{4}MTL-30%	0.740	0.810	0.700	0.650	0.780	0.310	0.400
	S^{4}MTL-20%	0.680	0.790	0.290	0.630	0.630	0.290	0.410
	S^{4}MTL-10%	0.670	0.800	0.000	0.620	0.620	0.300	0.840
	S^{4}MTL-5%	0.630	0.730	0.410	0.590	0.600	0.360	0.710

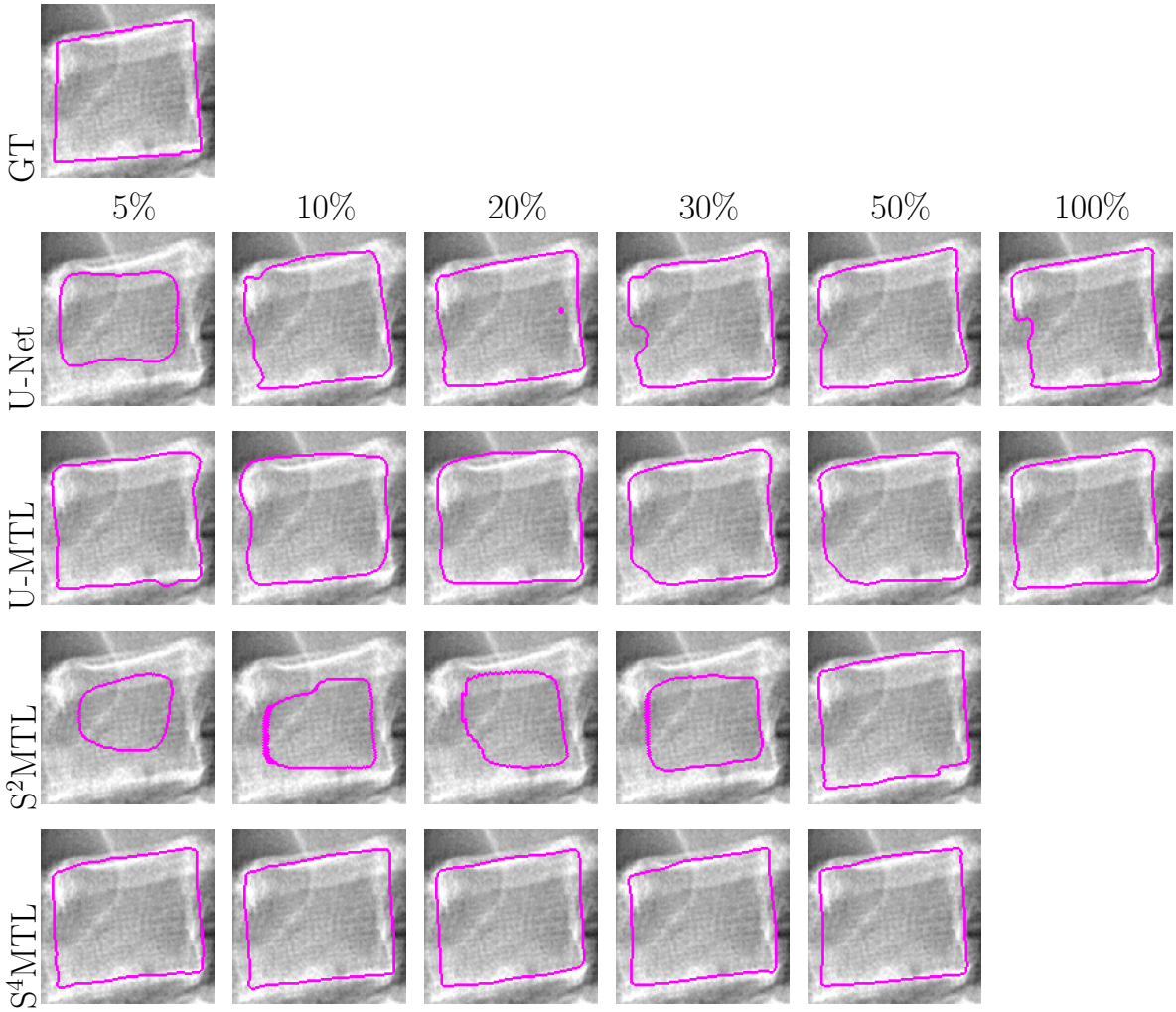


Figure 4.28: Spine Dataset: Boundary visualization of a predicted vertebra mask showing the superiority of our S^{4}MTL model over the baseline models with varying proportions of labeled data.

for different proportions of labeled data (Figure 4.25). Visualization of the segmented vertebra and lung boundaries by different models in varying labeled data proportions (Figure 4.28 and Figure 4.29) confirms the effectiveness of our model over competing models. For a fair comparison of all the models, a common segmentation architecture was used in the single-task for segmentation and in the multitask for segmentation mask generator. Our S^{4}MTL model consistently performs better than the semi-supervised and fully-supervised single-task segmentation (U-Net) and multitask (U-MTL) models, given the same number of labeled data during training ($|\mathcal{D}_{\mathcal{L}}|$). The advantage really comes with the knowledge gain from the larger proportion of unlabeled data and multitask learning,

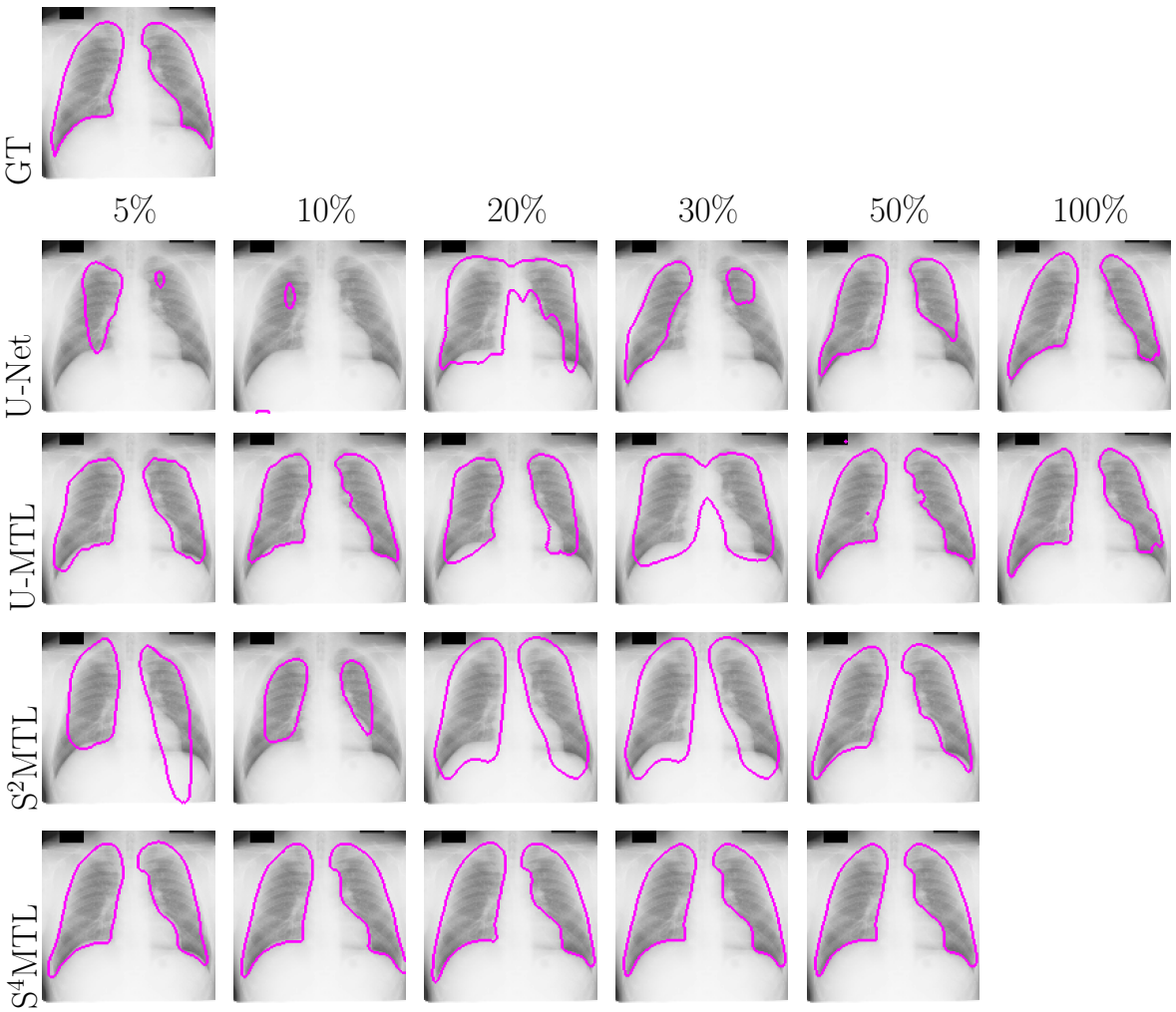


Figure 4.29: Chest Dataset: Boundary visualization of the predicted lung masks in a chest X-Ray shows consistent improvement with our S^{4}MTL model against the semi-supervised and fully-supervised baselines with varying proportions of labeled data.

S^{4}MTL's forte. This clearly reveals the effectiveness of our model.

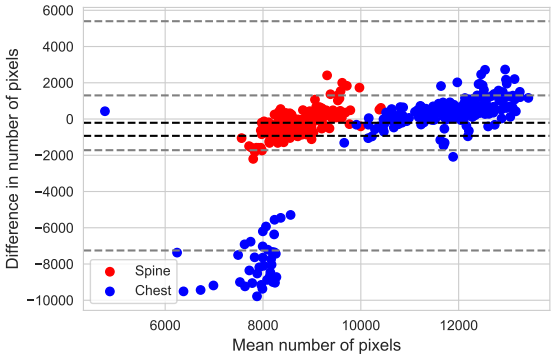
4.6.4 Statistical Analysis

To verify the significance of our model, we performed three different statistical tests, namely one-way ANOVA, paired-t, and Wilcoxon signed-rank tests. All three tests confirmed that our S^{4}MTL model with 50% labeled data model shows significant improvement over all the semi-supervised and fully-supervised baselines in the Chest Dataset (p-value < 0.05). Moreover, for the Spine Dataset, our S^{4}MTL model with 50% labeled data was found to

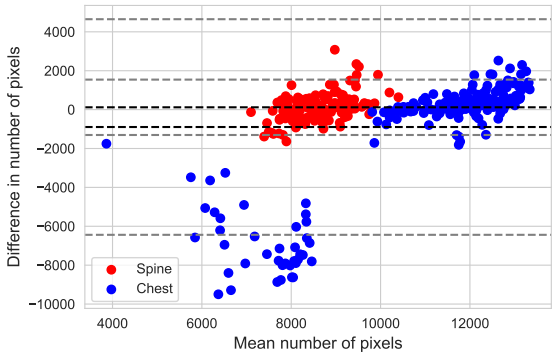
Table 4.23: Lung segmentation performance comparison of the S^{4}MTL model against the baselines for semi-supervised multitasking in different data settings with varying proportion of labeled chest X-ray image data.

Type	Model	DS	JI	SSIM	HD	Prec	Rec
Single-Task	U-Net-100%	0.922	0.856	0.816	4.524	0.909	0.936
	U-Net-50%	0.892	0.806	0.761	5.382	0.846	0.945
	U-Net-30%	0.834	0.715	0.654	6.584	0.723	0.986
	U-Net-20%	0.815	0.688	0.636	5.979	0.758	0.882
	U-Net-10%	0.800	0.667	0.599	7.111	0.682	0.969
	U-Net-5%	0.788	0.650	0.583	7.419	0.654	**0.991**
Multitask	U-MTL-100%	0.874	0.776	0.747	4.969	0.800	0.963
	U-MTL-50%	0.888	0.799	0.756	4.700	0.822	0.965
	U-MTL-30%	0.820	0.694	0.658	5.374	0.766	0.881
	U-MTL-20%	0.829	0.708	0.654	6.483	0.719	0.979
	U-MTL-10%	0.841	0.726	0.683	5.554	0.746	0.964
	U-MTL-5%	0.820	0.695	0.644	5.648	0.748	0.908
	S^{2}MTL-50%	0.810	0.681	0.636	4.862	0.782	0.840
	S^{2}MTL-30%	0.572	0.401	0.465	5.416	0.824	0.438
	S^{2}MTL-20%	0.607	0.436	0.516	5.782	0.987	0.438
	S^{2}MTL-10%	0.558	0.387	0.487	5.827	0.998	0.387
	S^{2}MTL-5%	0.458	0.300	0.421	7.496	0.989	0.300
	S^{4}MTL-50%	**0.946**	**0.898**	**0.864**	**3.432**	**0.964**	0.939
	S^{4}MTL-30%	0.903	0.823	0.798	3.963	0.885	0.921
	S^{4}MTL-20%	0.895	0.810	0.784	4.119	0.878	0.912
	S^{4}MTL-10%	0.868	0.768	0.742	4.272	0.834	0.905
	S^{4}MTL-5%	0.845	0.732	0.703	4.619	0.811	0.883

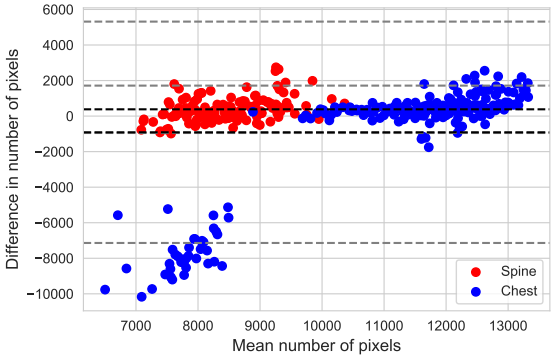
be significantly different from all the baseline models except the fully-supervised U-Net model. The Bland-Altman plots shown in Figure 4.30 also suggest good agreement between ground truth and the prediction by our S^{4}MTL models for both datasets. Moreover, we performed a robustness analysis to compare the class-wise segmentation performance of our S^{4}MTL model. Figure 4.31 shows that there are no significant differences among the class-wise segmentation performances (Dice scores). Furthermore, by performing one-way ANOVA, independent sample t-test, and Pearson correlation, we confirmed that our model is robust against different disease classes (normal vs abnormal in the Spine Dataset and normal vs TB vs nodule in the Chest Dataset).



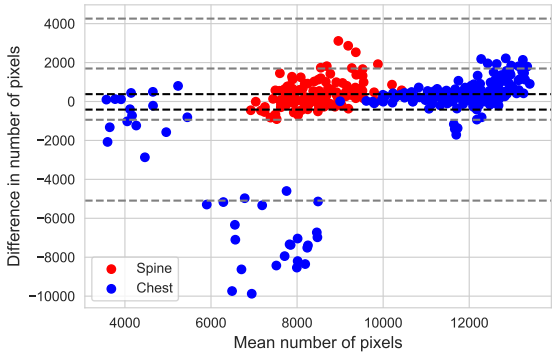
(a) S^{4}MTL-5%



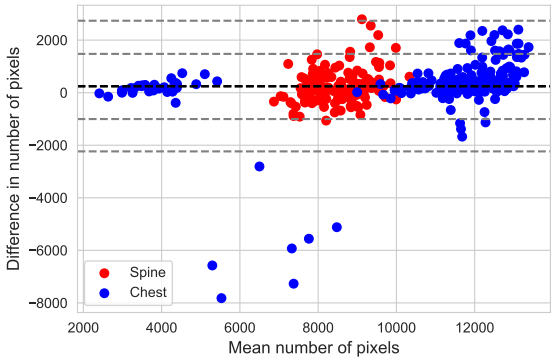
(b) S^{4}MTL-10%



(c) S^{4}MTL-20%



(d) S^{4}MTL-30%



(e) S^{4}MTL-50%

Figure 4.30: Bland-Altman plots show good agreement between the number of ground truth pixels and the different S^{4}MTL models-predicted pixels for the test sets.

Table 4.24: Vertebrae egmentation performance comparison of the S^{4}MTL model against the baselines for semi-supervised multitasking in different data settings with varying proportion of labeled spine X-ray patch data.

Type	Model	DS	JI	SSIM	HD	Prec	Rec
Single-Task	U-Net-100%	0.931	0.872	0.0.74	4.335	0.954	0.911
	U-Net-50%	0.919	0.851	0.857	4.569	0.949	0.893
	U-Net-30%	0.910	0.835	0.847	4.836	0.946	0.877
	U-Net-20%	0.903	0.824	0.839	5.022	0.935	0.873
	U-Net-10%	0.874	0.776	0.801	5.276	0.923	0.830
	U-Net-5%	0.702	0.541	0.685	6.909	0.957	0.554
Multitask	U-MTL-100%	0.888	0.799	0.817	5.348	0.908	0.869
	U-MTL-50%	0.890	0.802	0.827	5.429	0.936	0.849
	U-MTL-30%	0.881	0.787	0.817	5.421	0.950	0.821
	U-MTL-20%	0.862	0.758	0.792	5.733	0.902	0.826
	U-MTL-10%	0.873	0.775	0.804	6.396	0.884	0.863
	U-MTL-5%	0.856	0.333	0.584	7.018	**0.999**	0.333
	S^{2}MTL-50%	0.889	0.801	0.821	4.956	0.887	0.893
	S^{2}MTL-30%	0.752	0.603	0.719	6.032	0.997	0.603
	S^{2}MTL-20%	0.672	0.506	0.670	6.256	0.998	0.506
	S^{2}MTL-10%	0.640	0.471	0.654	6.772	0.998	0.471
	S^{2}MTL-5%	0.500	0.333	0.584	7.018	**0.999**	0.333
	S^{4}MTL-50%	**0.934**	**0.876**	**0.875**	**3.762**	0.947	**0.921**
	S^{4}MTL-30%	0.925	0.861	0.866	3.912	0.946	0.905
	S^{4}MTL-20%	0.921	0.853	0.860	4.162	0.942	0.900
	S^{4}MTL-10%	0.907	0.830	0.842	4.723	0.914	0.901
	S^{4}MTL-5%	0.890	0.802	0.821	4.835	0.887	0.893

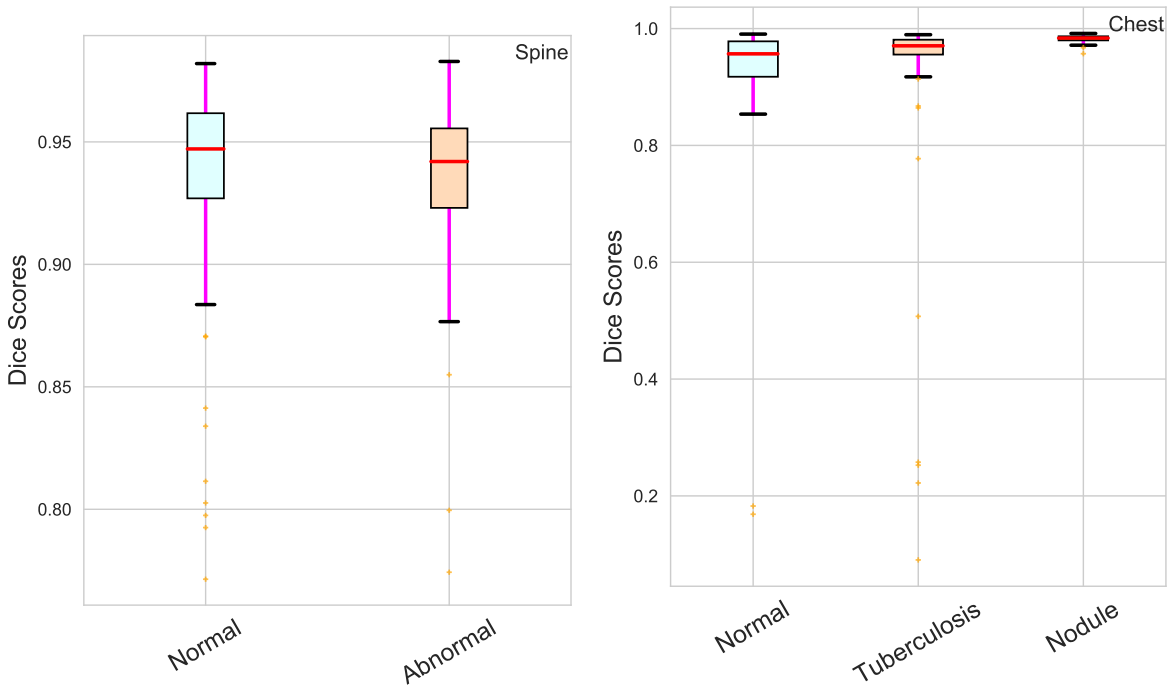


Figure 4.31: Class-wise boxplots showing the robustness of our S^{4}MTL-50% model in segmenting vertebrae and lungs from spinal and chest X-Rays.

CHAPTER 5

Conclusions and Future Research Directions

With the advent of deep learning techniques, learning-based medical image analysis is being vigorously explored. However, training deep neural networks in a fully supervised manner poses challenges due to the limited availability of large labeled datasets, which are difficult to obtain because of privacy issues, varying imaging configurations, disease types and severity, expensive and laborious manual annotation for ground truth, etc. Moreover, due to small sample sizes, learning algorithms can suffer from over-fitting, hindering the generalization across different datasets with disparate distributions, and inducing large domain shift. Confronting these challenges, we have developed and demonstrated a number of deep learning-based solutions for medical image analysis ranging from fully-supervised single-task to sparingly-supervised multi-task learning models. The latter solutions limit supervision to a small portion of labeled training samples and leverage a larger quantity of unlabeled samples, gaining additional knowledge using deep generative modeling, adversarial learning, and multi-task learning.

First, we demonstrated fully automatic 3D lung lobe segmentation method, exploiting progressive side-supervision in a dense V-Net (PDV-Net). Reliable and automatic lung lobe segmentation is a challenging task, especially in the presence of pathologies and incomplete fissures. We applied our PDV-Net to the automatic, fast, and reliable segmentation of lung lobes from chest CT scans. We evaluated our method using three test datasets—84 cases from LIDC, 154 cases from LTRC, and 55 cases from LOLA11. Our results demonstrated that our model outperforms, or at worst performs comparably to, the state-of-the-art while running at an average speed of 2 seconds per case, without requiring any prior segmentation. Furthermore, we demonstrated the robustness of our method against

varying configurations of CT reconstruction, choice of CT imaging device vendor, and the presence of lung pathologies. Including pathological and challenging cases in the training set could further improve the segmentation performance, especially, in the case of large domain shifts.

Second, we established a new state-of-the-art in fully automatic vertebrae segmentation in spinal X-Ray images using the progressive U-Net (PU-Net). The accurate and reliable segmentation of vertebrae is a prerequisite for the effective measurement of scoliosis. Our novel framework for accurately assessing scoliosis from anterior-posterior spine radiographs makes use of an end-to-end model that accurately and reliably segments spinal vertebrae, outputting a vertebrae segmentation mask that enables the accurate measurement of scoliosis through the calculation of the Cobb angle. Our pipeline promises to be an effective tool for the clinical diagnosis of scoliosis as well as for decision support in treatment planning. We envision combining the measurement of scoliosis with the training phase such that our model can make more intelligent predictions.

Third, we introduced progressive adversarial semantic segmentation (PASS), a novel semantic segmentation model for intelligently mitigating the domain shift problem caused by small datasets. We evaluated PASS using 8 public datasets for the tasks of retinal vessel segmentation from diabetic retinopathy images and lung segmentation from chest X-ray images. Our experimental results demonstrated the effectiveness of PASS in both in-domain and cross-dataset evaluations, even with smaller sample size and larger domain shift. An interesting next step would be to verify the effectiveness of PASS in iterative and active learning scenarios.

Fourth, we demonstrated the advantages of an ensemble of discriminators in the adversarial learning of variational autoencoders and applied this idea to semi-supervised classification from limited labeled data. Training our new multi-adversarial variational autoencoder network (MAVEN) models on small, labeled datasets and leveraging a large number of unlabeled training examples, we have shown superior performance relative to prior GAN and VAE-GAN based classifiers, suggesting that our MAVEN models can be very effective in concurrently generating high-quality realistic images and improving

multiclass classification performance. Furthermore, unlike conventional GAN-based semi-supervised classification, improvements in the classification of natural and medical images do not compromise the quality of the generated images. However, it remains an open problem to find the optimal number of discriminators that can perform consistently. Moreover, through conditional generation, images of rarer medical conditions could be augmented to obtain balanced training data, resulting in more effective classifier.

Fifth, we proposed and demonstrated the performance of a novel semi-supervised multi-task learning model for combined classification and segmentation from limited labeled chest X-ray images acquired in different settings. Experimental results showed our adversarial pyramid progressive attention U-Net (APPAU-Net) model to be competitive even against the single-task learning of fully-supervised models. Our future work will include more extensive experimentation of our model with larger numbers of classes and/or segmentation labels and larger quantities of unlabeled examples.

Lastly, incorporating a self-supervision branch, we introduced a novel self-supervised, semi-supervised, multitask learning (S^4MTL) model and validated it in combined medical image classification and segmentation experiments with limited labeled data. The effectiveness of our S^4MTL model over semi-supervised and fully-supervised single-task and multitask models was confirmed by our experimental results. Moreover, S^4MTL with 50% labeled data achieved better performance, with statistical significance, over all other semi-supervised models. Our model may be utilized in general settings when data available with and without labels have dissimilar distributions. A worthwhile next step would be to experiment with more sophisticated medical imaging and computer vision data at larger scale and in more challenging settings.

APPENDIX A

Datasets

The following datasests were used in the experiments, in order to validate the models and algorithms presented in Chapter 3.

- **LIDC:** A subset of chest CT volumes (354 cases) from the LIDC dataset (Armato et al., 2011) was selected for annotation. To ensure variation in the data, the CT scans were selected such that both challenging and visible fissures are well-represented in the dataset. The lobe segmentation ground truth masks were generated in a semi-automatic fashion by multiple human annotators using the chest imaging platform feature of 3D Slicer. To mitigate bias in the ground truth, the generated masks were later refined and validated by an expert radiologist. The dataset was split into 270 training and 84 test cases. 10% of the training set was utilized as the validation set to select values for the hyper-parameters. The CT scans used in the experiment have a variable number of slices with each CT volume containing 100 to 672 slices of size 512×512 pixels. Figure A.1 shows the histograms of the number of slices per volume, and of the voxel dimensions which vary between 0.49–0.98 mm, 0.49–0.98 mm, and 0.45–3.00 mm along the x, y, and z axes, respectively. Therefore, the selected CT scans used for pulmonary lobe segmentation not only exhibit varying shapes of fissures and lobes, but also show a variable number of slices and voxel sizes.

- **LTRC:** 154 CT scans were selected from the LTRC database (Karwoski et al., 2008). The LTRC dataset includes lobe masks for pathological cases that have clear evidence of chronic obstructive pulmonary disease (COPD) or interstitial lung

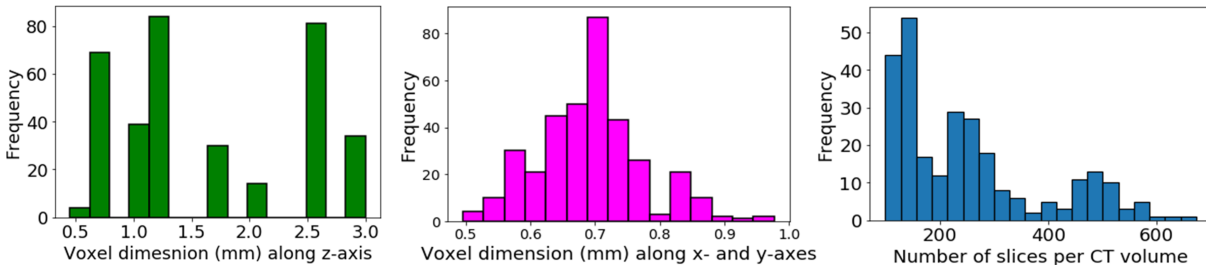


Figure A.1: Histograms (from left) of the number of slices per volume; voxel dimensions along the x and y axes; and voxel dimensions along the z axis of lung CT scans in the entire LIDC dataset.

disease (ILD), including emphysema and fibrosis. The LTRC cases allow us to measure the robustness of our model against pathologies in the lungs.

- **LOLA11:** Lobe and Lung Analysis (LOLA11) challenge provides a data set of 55 chest CT scans with varying abnormalities (LOLA11, 2011). Any algorithm can be evaluated on LOLA11 dataset upon submission of the results compared against established reference standards of lung and lobe segmentations. The CT scans are the representative of a variety of clinically common scanners and protocols. Based on the performance of initial methods for lung/lobe segmentation, the scans for the challenge were selected in a way so that there is a balance between easy and hard scans. The maximum slice spacing present in the scans is 1.5mm, where most of them are isotropic.

- **APSeg:** APSeg is a dataset of 100 high-resolution anterior-posterior spine X-Ray images of children with evidence of scoliosis in various extents (Imran et al., 2019c). The dataset includes expert-provided segmentation annotation of 18 relevant vertebrae (cervical C7, thoracic T1–T12, lumbar L1–L5) in each of the X-rays. The dataset is split into training (80), testing (15), and validation (5) sets.

- **SVHN:** The Street View House Numbers (SVHN) dataset (Netzer et al., 2011) (Figure A.2). There are 73,257 digit images for training and 26,032 digit images for testing. Out of two versions of the images, we used the version which has MNIST-like 32×32 pixel RGB color images centered around a single digit. Each

image is labeled as belonging to one of 10 classes: digits 0–9.

- **CIFAR-10:** The CIFAR-10 dataset (Krizhevsky and Hinton, 2009) (Figure A.3). It consists of 60,000 32×32 pixel RGB color images in 10 classes. There are 50,000 training images and 10,000 test images. Each image is labeled as belonging to one of 10 classes: plane, auto, bird, cat, deer, dog, frog, horse, ship, and truck.

- **CXR:** The anterior-posterior Chest X-Ray (CXR) dataset (Kermany et al., 2018) (Figure A.4). The dataset contains 5,216 training and 624 test images. Each image is labeled as belonging to one of 3 classes: normal, bacterial pneumonia (b-pneumonia), and viral pneumonia (v-pneumonia).

- **SLC:** The skin lesion classification (SLC) dataset (Figure A.5). We employed 2,000 RGB skin images from the ISIC 2017 dermoscopy image dataset (Codella et al., 2018); of which we used 1,600 for training and 400 for testing. Each image is labeled as belonging to one of 2 classes: non-melanoma and melanoma.

- **MCU:** In the MCU dataset (Figure A.7a) (Shiraishi et al., 2000), there are 138 frontal X-Rays: 80 X-Rays are normal and 58 X-Rays with manifestations of Tuberculosis. This dataset contains separate left and right lung ground truth masks which are combined for binary segmentation of the lungs.

- **CHN:** CHN dataset has 662 frontal chest X-rays (Figure A.7b) (Shiraishi et al., 2000). Of them 336 normal X-rays and 326 abnormal cases with manifestations of TB, including pediatric X-rays. After carefully examining all the cases, we created two versions: 1) CHN-V1—566 X-rays which include 287 normal and 279 with abnormalities and 2) CHN-V2—527 X-rays based on good agreement with the corresponding ground truth lung masks which include 248 normal and 279 abnormal X-rays.

- **JSRT:** This database contains 247 chest X-rays (Figure A.7c) in which 154 images contain pulmonary lung nodule and 93 images contain no lung nodules (Jaeger et al., 2014). In addition to the lung masks (separated left-right), this dataset includes

ground truth masks for heart and clavicles (separated left-right). Based on split, we call two versions as JSRT-V1 and JSRT-V2.

- **Chest:** Combining the three publicly available chest X-Ray datasets (MCU, CHN-V2, JSRT), a dataset of 912 chest X-ray images is created which we call the "Chest" dataset (Imran and Terzopoulos, 2019b) (A.1). The dataset is split into three sets—training (615), validation (69), and test (228). Along with lung segmentation, the Chest dataset has 3-class annotations—normal, tuberculosis (TB), and nodule.

- **Spine:** We used a dataset of 100 very high resolution lateral spine X-ray images (Wong and McGirt, 2013) and corresponding ground truth segmentation masks of multi-class vertebrae annotations. Extending the individual vertebrae region from each of the X-ray images, we extracted vertebra patches. Figure A.6 illustrates the extraction of vertebra pathces from a lateral spine X-ray image. This patch extraction results in a dataset of 994 patch images (Imran et al., 2020c) including verterbra mask and osteoporotic class label for each of the vertebra pathces. This is named "Spine" dataset. The dataset was split into three subsets: train (713), validation (42), and test (139). We performed vertebra segmentation and abnormality prediction on each vertebra patch (normal vs abnormal).

- **ARIA:** ARIA dataset (available in the public domain [1]) contains 143 color fundus images (Figure A.8a) of adults, collected at St Paul's Eye Unit and the University of Liverpool, United Kingdom. The dataset includes blood vessel masks created by trained image analysis experts. Of them 61 images are healthy and 82 diseased fundoscopic images A.2.

- **CHASE:** The CHASE dataset (publicly available at [2]) contains 28 eye fundus images (Figure A.8b) with a resolution of 1280×960. Two sets of ground-truth vessel annotations are available in the dataset. The first is commonly used for

[1] https://sourceforge.net/projects/aria-vessels/

[2] https://blogs.kingston.ac.uk/retinal/chasedb1/

Figure A.2: Example images of each class in the SVHN digit image dataset. Classes L→R: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Figure A.3: Example images of each class in the CIFAR-10 image dataset. Classes L→R: plane, auto, bird, cat, deer, dog, frog, horse, ship, truck.

training and testing, while the second set refers to human baseline.

- **DRIVE:** Digital Retinal Images for Vessel Extraction (DRIVE) is available in the public domain [3]. DRIVE includes 40 randomly selected images (Figure A.8c) (33 without and 7 with diabetic retinopathy signs) from a population of 400 subjects. Each image is of 768 × 584 pixels.

- **HRF:** The High-Resolution Fundus (HRF) database contains 15 images of healthy patients, 15 images of patients with diabetic retinopathy and 15 images of glaucomatous patients (Figure A.8d). Binary gold standard vessel segmentation images generated by a group of experts working in the field of retinal image analysis and clinicians from the cooperated ophthalmology clinics, are available for each image. The HRF dataset is available at the public domain [4].

- **STARE:** STructured Analysis of the Retina (STARE) dataset is available in the public domain [5] (Figure A.8e). It consists of 10 healthy and 10 diseased images along with the corresponding vessel segmentation ground truth masks. The images are 605 × 700 pixels in resolution.

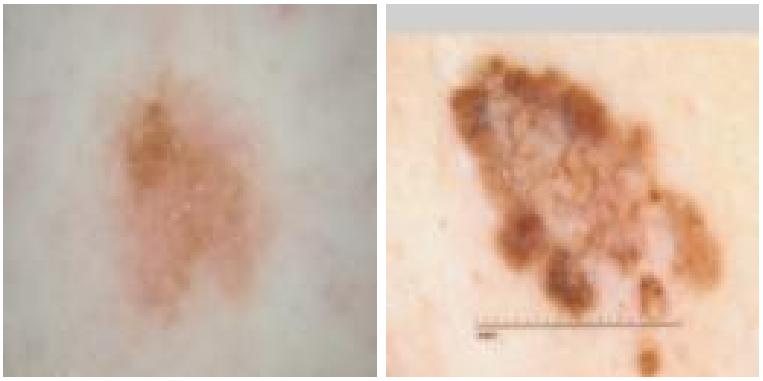
[3] http://www.isi.uu.nl/Research/Databases/DRIVE/

[4] https://www5.cs.fau.de/research/data/fundus-images/

[5] http://cecas.clemson.edu/~ahoover/stare/probing/

(a) Normal (b) b-Pneumonia (c) v-Pneumonia

Figure A.4: Examples images of each class in the CXR dataset.

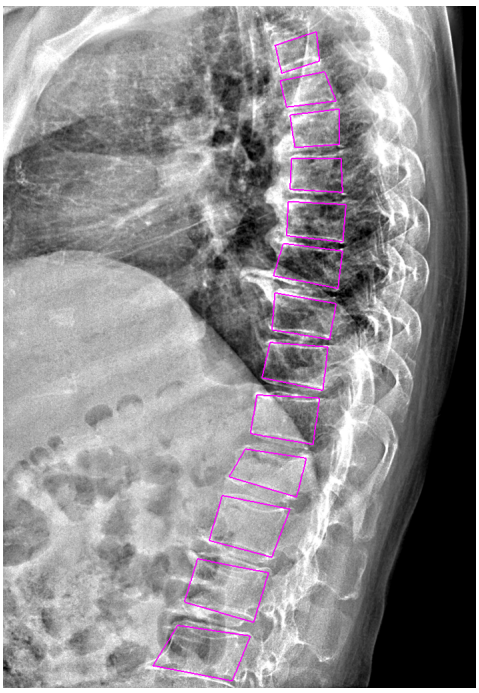


(a) Non-melanoma (b) Melanoma

Figure A.5: Example images of each class in the SLC datasets.

Table A.1: Split of the chest X-ray datasets for training, testing, and validation (for model selections).

Dataset	#Chest X-rays			Data splits			Abnormality Classes
	Total	Normal	Abnormal	Train	Val	Test	
MCU	138	80	58	93	10	35	TB
CHN-V1	566	279	287	381	43	142	normal, TB
CHN-V2	527	248	279	355	40	132	normal, TB
JSRT-V1	247	93	154	166	19	62	normal, Nodule
JSRT-V2	247	93	154	111	13	123	normal, Nodule
Chest	912	421	491	615	69	228	normal, TB, Nodule



Lateral spine X-Ray image

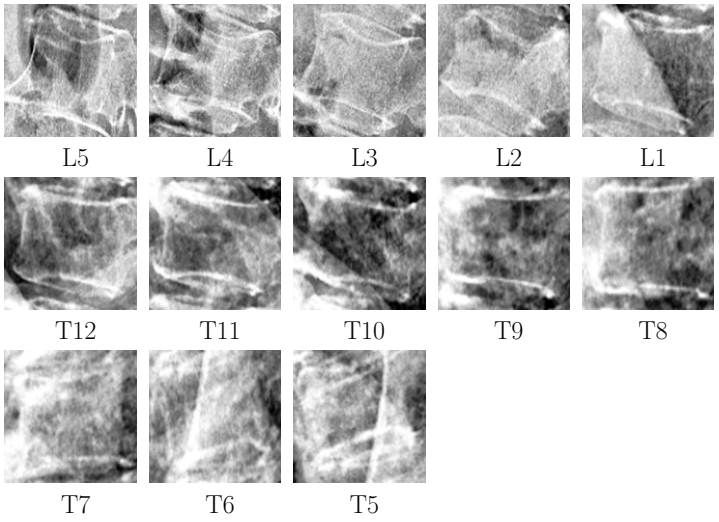


Figure A.6: Individual vertebra patches along with their labels (vertebra indices) extracted from the lateral X-Ray image. L: Lumbar vertebra, T: Thoracic vertebra. L5 is the bottom most vertebra, then other vertebrae are shown in reverse order. In this X-Ray, only L2 is labeled abnormal while the remaining vertebrae are normal.

Table A.2: Split of fundoscopic image datasets used in our experiments.

Dataset	#Fundus images			Data splits		
	Total	Healthy	Diseased	Train	Val	Test
DRIVE	40	33	7	18	2	20
STARE	20	10	10	10	2	8
CHASE	28	20	8	17	5	6
ARIA	143	61	82	121	5	17
HRF	45	15	30	26	5	14

(a) MCU (b) CHN (c) JSRT

Figure A.7: Example images from three different chest X-ray image datasets used for pulmonary segmentation.

(a) ARIA (b) CHASE (c) DRIVE (d) HRF (e) STARE

Figure A.8: Example images from five different diabetic retinopathy image datasets used for vascular segmentation.

Anitha, H. and Prabhu, G. (2012). Automatic quantification of spinal curvature in scoliotic radiograph using image processing. *Journal of Medical Systems*, 36(3):1943–1951. 20

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*. 24

Armato, S. G., McLennan, G., Bidaut, L., McNitt-Gray, M. F., Meyer, C. R., Reeves, A. P., Zhao, B., Aberle, D. R., Henschke, C. I., Hoffman, E. A., et al. (2011). The lung image database consortium (LIDC) and image database resource initiative (IDRI): A completed reference database of lung nodules on CT scans. *Medical physics*, 38(2):915–931. 111

Aziz, A., Ashizawa, K., Nagaoki, K., and Hayashi, K. (2004). High resolution CT anatomy of the pulmonary fissures. *J Thoracic Imag*, 19(3):186–191. 14

Bauer, C., Eberlein, M., and Beichel, R. (2018). Pulmonary lobe separation in expiration chest ct scans based on subject-specific priors derived from inspiration scans. *J of Med Imag*, 5(1):014003. 16

Beauchamp, M., Labelle, H., Grimard, G., Stanciu, C., Poitras, B., and Dansereau, J. (1993). Diurnal variation of cobb angle measurement in adolescent idiopathic scoliosis. *Spine*, 18(12):1581–1583. 20

Bermudez, C., Plassard, A. J., Davis, L. T., Newton, A. T., Resnick, S. M., and Landman, B. A. (2018). Learning implicit brain mri manifolds with deep learning. In *Medical Imaging 2018: Image Processing*, volume 10574, page 105741L. International Society for Optics and Photonics. 24

Blaffert, T., Wiemker, R., Barschdorf, H., Kabus, S., Klinder, T., Lorenz, C., Schadewaldt, N., and Dharaiya, E. (2010). A completely automated processing pipeline for lung and lung lobe segmentation and its application to the lidc-idri data base. In *Medical Imaging 2010: Image Processing*, volume 7623, page 762347. International Society for Optics and Photonics. 16

Bragman, F. J., McClelland, J. R., Jacob, J., Hurst, J. R., and Hawkes, D. J. (2017). Pulmonary lobe segmentation with probabilistic segmentation of the fissures and a groupwise fissure prior. *IEEE Tran on Med Imag*, 36(8):1650–1663. 15, 63

Bulatov, Y. (2018). Saving memory using gradient-checkpointing. 59

Calimeri, F., Marzullo, A., Stamile, C., and Terracina, G. (2017). Biomedical data augmentation using generative adversarial neural networks. In *International Conference on Artificial Neural Networks*, pages 626–634. Springer. 24

Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In *International Conf. on Machine Learning*. 26

Cassar-Pullicino, V. and Eisenstein, S. (2002). Imaging in scoliosis: what, why and how? *Clinical radiology*, 57(7):543–562. 68

Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542. 4

Chen, C., Dou, Q., Chen, H., Qin, J., and Heng, P.-A. (2019). Synergistic image and feature adaptation: Towards cross-modality domain adaptation for medical image segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 865–872. 25

Chen, C., Dou, Q., et al. (2018). Semantic-aware generative adversarial nets for unsupervised domain adaptation in chest X-ray segment. *arXiv:1806.00600*. 22, 85

Chowanska, J., Kotwicki, T., Rosadzinski, K., and Sliwinski, Z. (2012). School screening for scoliosis: Can surface topography replace examination with scoliometer? *Scoliosis*, 7(1):9. 69

Chuquicusma, M. J., Hussein, S., Burt, J., and Bagci, U. (2018). How to fool radiologists with generative adversarial networks? a visual turing test for lung cancer diagnosis. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 240–244. IEEE. 24

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer. 11

Ciresan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*. 11

Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., et al. (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172. IEEE. 113

Dai, W., Dong, N., Wang, Z., Liang, X., Zhang, H., and Xing, E. P. (2018). Scan: Structure correcting adversarial network for organ segmentation in chest x-rays. In *Deep Learning in Medical Image Analysis*, volume 11045 of *LNCS*. Springer. 22

Denton, E. L., Chintala, S., Szlam, A., and Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*. 24

Doel, T., Matin, T., Gleeson, F., Gavaghan, D., and Grau, V. (2012). Pulmonary lobe segmentation from CT images using fissureness, airways, vessels, and multilevel b-splines. *Proc. of ISBI*, 9:1491–1494. 15

Donahue, J., Krähenbühl, P., and Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*. 22

Dong, N., Kampffmeyer, M., Liang, X., Wang, Z., Dai, W., and Xing, E. (2018). Unsupervised domain adaptation for automatic estimation of cardiothoracic ratio. In *International conference on medical image computing and computer-assisted intervention*, pages 544–552. Springer. 84

Dou, Q., Ouyang, C., Chen, C., Chen, H., Glocker, B., Zhuang, X., and Heng, P.-A. (2018). Pnp-adanet: Plug-and-play adversarial domain adaptation network with a benchmark at cross-modality cardiac segmentation. *arXiv preprint arXiv:1812.07907*. 25

Durugkar, I., Gemp, I., and Mahadevan, S. (2016). Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*. 23, 35

Ferreira, F. T., Sousa, P., Galdran, A., Sousa, M. R., and Campilho, A. (2018). End-to-end supervised lung lobe segmentation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE. 17

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *arXiv preprint arXiv:1803.01229*. 24

Fu, H., Cheng, J., Xu, Y., et al. (2018). Joint optic disc and cup segmentation based on multi-label deep network and polar transformation. *IEEE TMI*. 48, 84, 85

George, K., Harrison, A., Jin, D., Xu, Z., and Mollura, D. (2017). Pathological pulmonary lobe segmentation from CT images using progressive holistically nested neural networks and random walker. *DLMIA. LNCS*, page 10553. 17, 59, 62, 66

Gibson, E., Giganti, F., Hu, Y., Bonmati, E., Bandula, S., Gurusamy, K., Davidson, B., Pereira, S. P., Clarkson, M. J., and Barratt, D. C. (2018). Automatic multi-organ segmentation on abdominal CT with dense V-networks. *IEEE Tran on Med Imag*. 12, 28, 59

Girard, F., Kavalec, C., and Cheriet, F. (2019). Joint segmentation and classification of retinal arteries/veins from fundus images. *AI in Med*. 26

Giuliani, N., Payer, C., Pienn, M., Olschewski, H., and Urschler, M. (2018). Pulmonary lobe segmentation in ct image using alpha-expansion. *Proc. of VISIGRAPP*, pages 387–394. 15, 63

Gonzalez, R. C., Woods, R. E., and Eddins, S. L. (2004). *Digital image processing using MATLAB*. Pearson Education India. 10

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680. 22

Gu, Z., Cheng, J., Fu, H., Zhou, K., Hao, H., Zhao, Y., Zhang, T., Gao, S., and Liu, J. (2019). Ce-net: context encoder network for 2d medical image segmentation. *IEEE transactions on medical imaging*, 38(10):2281–2292. 11

Guan, Q., Huang, Y., et al. (2018). Diagnose like a radiologist: Attention guided convolutional net for thorax disease classification. *arXiv:1801.09927*. 22

Guibas, J. T., Virdi, T. S., and Li, P. S. (2017). Synthetic medical images from dual generative adversarial networks. *arXiv preprint arXiv:1709.01872*. 24

Gulsun, M., Ariyurek, O., Comert, R., and Karabulut, N. (2006). Variability of the pulmonary oblique fissures presented by high-resolution computed tomography. *Surgical Radilogic Anatomy*, 28(3):293–299. 14

Han, C., Hayashi, H., Rundo, L., Araki, R., Shimoda, W., Muramatsu, S., Furukawa, Y., Mauri, G., and Nakayama, H. (2018). Gan-based synthetic brain mr image generation. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 734–738. IEEE. 24

Harrison, A. P., Xu, Z., George, K., Lu, L., Summers, R. M., and Mollura, D. J. (2017). Progressive and multi-path holistically nested neural networks for pathological lung segmentation from CT images. In *Proc. of MICCAI*, pages 621–629. 28

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637. 72, 74

Horng, M.-H., Kuok, C.-P., Fu, M.-J., Lin, C.-J., and Sun, Y.-N. (2019). Cobb angle measurement of spine from x-ray images using convolutional neural network. *Computational and Mathematical Methods in Medicine*, 2019. 20, 67, 69

Hou, L., Agarwal, A., Samaras, D., Kurc, T. M., Gupta, R. R., and Saltz, J. H. (2017). Unsupervised histopathology image synthesis. *arXiv preprint arXiv:1712.05021*. 24

Huo, Y., Xu, Z., Moon, H., Bao, S., Assad, A., Moyo, T. K., Savona, M. R., Abramson, R. G., and Landman, B. A. (2018). Synseg-net: Synthetic segmentation without target modality ground truth. *IEEE transactions on medical imaging*, 38(4):1016–1025. 25

Imran, A.-A.-Z., Bakic, P. R., Maidment, A. D., and Pokrajac, D. D. (2017). Optimization of the simulation parameters for improving realism in anthropomorphic breast phantoms. *Proceedings of the SPIE, Volume 10132, id. 1013257 7*, 132:1315–1324. 73

Imran, A.-A.-Z., Hatamizadeh, A., Ananth, S. P., Ding, X., Tajbakhsh, N., and Terzopoulos, D. (2019a). Fast and automatic segmentation of pulmonary lobes from chest ct using a progressive dense V-network. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pages 1–10. 6, 12, 17, 31

Imran, A.-A.-Z., Hatamizadeh, A., Ananth, S. P., Ding, X., Terzopoulos, D., and Tajbakhsh, N. (2018). Automatic segmentation of pulmonary lobes using a progressive dense V-network. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, volume 11045 of *Lecture Notes in Computer Science*, pages 282–290. Springer. 6, 12, 17, 31, 42, 48

Imran, A.-A.-Z., Huang, C., Tang, H., Fan, W., Cheung, K. M., To, M., Qian, Z., and Terzopoulos, D. (2019b). Bipartite distance for shape-aware landmark detection in spinal X-ray images. In *Medical Imaging Meets NeurIPS Workshop*, Vancouver, Canada. 20

Imran, A.-A.-Z., Huang, C., Tang, H., Fan, W., Cheung, K. M., To, M., Qian, Z., and Terzopoulos, D. (2019c). End-to-end fully automatic segmentation of scoliotic vertebrae in spinal X-ray images. In *Medical Imaging Meets NeurIPS Workshop*, Vancouver, Canada. 6, 18, 84, 85, 112

Imran, A.-A.-Z., Huang, C., Tang, H., Fan, W., Cheung, K. M., To, M., Qian, Z., and Terzopoulos, D. (2020a). Analysis of scoliosis from spinal x-ray images. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*. 6, 18

Imran, A.-A.-Z., Huang, C., Tang, H., Fan, W., Xiao, Y., Hao, D., Qian, Z., and Terzopoulos, D. (2020b). Partly supervised multitask learning. *arXiv preprint arXiv:2005.02523*. 27

Imran, A.-A.-Z., Huang, C., Tang, H., Fan, W., Xiao, Y., Hao, D., Qian, Z., and Terzopoulos, D. (2020c). Self-supervised, semi-supervised, multi-context learning for the combined classification and segmentation of medical images. In *34th AAAI Conference on Artificial Intelligence*, New York, USA. 8, 114

Imran, A.-A.-Z. and Terzopoulos, D. (2019a). Multi-adversarial variational autoencoder networks. *arXiv preprint arXiv:1906.06430*. 7, 22, 24

Imran, A.-A.-Z. and Terzopoulos, D. (2019b). Semi-supervised multi-task learning with chest X-ray images. In *Machine Learning in Medical Imaging*, volume 11861 of *Lecture Notes in Computer Science*, pages 151–159. Springer. 7, 22, 24, 26, 56, 84, 85, 97, 114

Imran, A.-A.-Z. and Terzopoulos, D. (2019c). Semi-supervised multi-task learning with chest X-ray images. *arXiv preprint arXiv:1908.03693*. 27

Imran, A.-A.-Z. and Terzopoulos, D. (2020). Progressive adversarial semantic segmentation. *arXiv preprint arXiv:2005.04311*. 7, 26

Imran, A.-A.-Z. and Terzopoulos, D. (2021). Multi-adversarial variational autoencoder nets for simultaneous image generation and classification. In *Deep Learning Applications, Volume 2*. Springer. 7, 24

Iwano, S., Kitano, M., Matsuo, K., Kawakami, K., Koike, W., Kishimoto, M., Inoue, T., Li, Y., and Naganawa, S. (2013). Pulmonary lobar volumetry using novel volumetric

computer-aided diagnosis and computed tomography. *Interactive cardiovascular and thoracic surgery*, 17(1):59–65. 15

Izadi, S., Mirikharaji, Z., Kawahara, J., and Hamarneh, G. (2018). Generative adversarial networks to segment skin lesions. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 881–884. IEEE. 84, 85

Jaeger, S., Candemir, S., et al. (2014). Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. *Quant Imag in Med and Surg*. 113

Kamnitsas, K., Ledig, C., Newcombe, V. F., Simpson, J. P., Kane, A. D., Menon, D. K., Rueckert, D., and Glocker, B. (2017). Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78. 11, 25

Karwoski, R. A., Bartholmai, B., Zavaletta, V. A., Holmes, D., and Robb, R. A. (2008). Processing of ct images for analysis of diffuse lung disease in the lung tissue research consortium. In *Medical Imaging 2008: Physiology, Function, and Structure from Medical Images*, volume 6916, page 691614. International Society for Optics and Photonics. 111

Kawchuk, G. and McArthur, R. (1997). Scoliosis quantification: an overview. *The Journal of the Canadian Chiropractic Association*, 41(3):137. 20

Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., et al. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131. 113

Khosravan, N. and Bagci, U. (2018). Semi-supervised multi-task learning for lung cancer diagnosis. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 710–713. IEEE. 26

Kim, H., Kim, H. S., Moon, E. S., Yoon, C.-S., Chung, T.-S., Song, H.-T., Suh, J.-S., Lee, Y. H., and Kim, S. (2010). Scoliosis imaging: what radiologists should know. *Radiographics*, 30(7):1823–1842. 18, 20

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*. 60

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. 22

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer. 23, 113

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105. 11

Kusuma, B. A. (2017). Determination of spinal curvature from scoliosis X-ray images using K-means and curve fitting for early detection of scoliosis disease. *ICITISEE*. 20, 69

Larrazabal, A. J., Martinez, C., and Ferrante, E. (2019). Anatomical priors for image segmentation via post-processing with denoising autoencoders. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 585–593. Springer. 25

Lassen, B. and van Rikxoort, E. (2013). Automatic segmentation of the pulmonary lobes from chest ct scans based on fissures, vessels, and bronchi. *IEEE Tran on Med Imag*, 32(2):210–222. 15

Lassen-Schmidt, B., Kuhnigk, J., Konrad, O., Van Ginneken, B., and Van Rikxoort, E. (2017). Fast interactive segmentation of the pulmonary lobes from thoracic computed tomography data. *Physics in Medicine & Biology*, 62(16):6649. 16

LeCun, Y., Kavukcuoglu, K., Farabet, C., et al. (2010). Convolutional networks and applications in vision. In *International Symposium on Circuits and Systems (ISCAS'10)*. 53

Lessmann, N., van Ginneken, B., de Jong, P. A., and Išgum, I. (2019). Iterative fully convolutional neural networks for automatic vertebra segmentation and identification. *Medical Image Analysis*, 53:142–155. 20

Li, X., Chen, H., Qi, X., Dou, Q., Fu, C., and Heng, P. (2018). H-denseunet: Hybrid densely connected unet for liver and tumor segmentation from ct volumes. *IEEE Transactions on Medical Imaging*, 37(12):2663–2674. 12

Liebel, L. and Körner, M. (2018). Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*. 26

Lim, H.-j., Weinheimer, O., Wielpütz, M. O., Dinkel, J., Hielscher, T., Gompelmann, D., Kauczor, H.-U., and Heussel, C. P. (2016). Fully automated pulmonary lobar segmentation: influence of different prototype software programs onto quantitative evaluation of chronic obstructive lung disease. *PLoS One*, 11(3):e0151498. 16

Liu, X. et al. (2019). Joint learning of multiple image restoration tasks. *CoRR*. 26

LOLA11 (2011). Lobe and lung analysis. 17, 112

Long, J., Shelhamer, E., and Darrell, T. (2014). Fully convolutional networks for semantic segmentation. 11

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press. 3

Madani, A. et al. (2018). Semi-supervised learning with GANs for chest X-ray classification with ability of data domain adaptation. In *Proc. ISBI*. 22, 24

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*. 35

Mateusiak, M. and Mikolajczyk, K. (2019). Semi-automatic spine segmentation method of ct data. In *Mechatronics 2019: Recent Advances Towards Industry 4.0*. 20

Mehra, J. and Neeru, N. (2016). A brief review: Super-pixel based image segmentation methods. *Imperial journal of interdisciplinary research*, 2. 11

Mehta, S., Mercan, E., Bartlett, J., et al. (2018). Y-Net: Joint segmentation and classification for diagnosis of breast biopsy images. In *MICCAI*. 26, 98

Milletari, F., Navab, N., and Ahmadi, S. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Proc. of 3DV*, pages 565–571. IEEE. 11, 30

Mittal, A., Hooda, R., and Sofat, S. (2018). Lf-segnet: A fully convolutional encoder–decoder network for segmenting lung fields from chest radiographs. *WPC*. 96

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*. 74

Mordido, G., Yang, H., and Meinel, C. (2018). Dropout-gan: Learning from a dynamic ensemble of discriminators. *arXiv preprint arXiv:1807.11346*. 23, 35, 74

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, page 5. 23, 112

Neyshabur, B., Bhojanapalli, S., and Chakrabarti, A. (2017). Stabilizing gan training with multiple random projections. *arXiv preprint arXiv:1705.07831*. 23

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848. 3

Nguyen, T., Le, T., Vu, H., and Phung, D. (2017). Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2670–2680. 23

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*. 24

Oktay, O., Schlemper, J., Folgoc, L. L., et al. (2018). Attention U-net: Learning where to look for the pancreas. *arXiv:1804.03999*. 12, 49, 84, 85

Oliveira, H., Ferreira, E., and Santos, J. A. d. (2019). Truly generalizable radiograph segmentation with conditional domain adaptation. *arXiv preprint arXiv:1901.05553*. 85

Ostrovski, G., Dabney, W., and Munos, R. (2018). Autoregressive quantile networks for generative modeling. *arXiv preprint arXiv:1806.05575*. 74

Pruijs, J., Hageman, M., Keessen, W., Van Der Meer, R., and Van Wieringen, J. (1994). Variation in cobb angle measurements in scoliosis. *Skeletal radiology*, 23(7):517–520. 20

Pu, J., Zheng, B., Leader, J., Fuhrman, C., Knollmann, F., Klym, A., and Gur, D. (2009). Pulmonary lobe segmentation in ct examinations using implicit surface fitting. *IEEE Tran on Med Imag*, 28(12):1986–1996. 17

Qadri, S. F., Zhao, Z., Ai, D., Ahmad, M., and Wang, Y. (2019). Vertebrae segmentation via stacked sparse autoencoder from computed tomography images. In *Eleventh International Conference on Digital Image Processing (ICDIP 2019)*, volume 11179, page 111794K. International Society for Optics and Photonics. 20

Raasch, B., Carsky, E., Lane, E., O'Callaghan, J., and Heitzman, E. (1982). Radiographic anatomy of the interlobar fissures: A study of 100 specimens. *Am J Roentgenol*, 138(6):1043–1049. 13

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*. 24, 69

Ravuri, S., Mohamed, S., Rosca, M., and Vinyals, O. (2018). Learning implicit generative models with the method of learned moments. *arXiv preprint arXiv:1806.11006*. 74

Rezaei, M., Yang, H., et al. (2018). Multi-task generative adversarial network for handling imbalanced clinical data. *CoRR*. 26

Ronneberger, O., Fischer, P., and Brox, T. (2015a). U-net: Convolutional networks for biomedical image segmentation. In *Proc. of MICCAI*, pages 234–241. Springer. 11, 59

Ronneberger, O., Fischer, P., and Brox, T. (2015b). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer. 31, 42, 53, 84, 85, 97

Ross, J., Estépar, R. S. J., Kindlmann, G., Díaz, A., Westin, C., Silverman, E., and Washko, G. (2010). Automatic lobe segmentation using particles, thin plate splines, and a maximum a posteriori estimation. *Proc. of MICCAI*, 6363:163–171. 16

Roth, H. R., Oda, H., Hayashi, Y., Oda, M., Shimizu, N., Fujiwara, M., Misawa, K., and Mori, K. (2017). Hierarchical 3d fully convolutional networks for multi-organ segmentation. 12

Rottman, B. M. and Hastie, R. (2014). Reasoning about causal relationships: Inferences on causal networks. *Psychological bulletin*, 140(1):109. 3

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*. 26

Salehinejad, H., Valaee, S., Dowdell, T., Colak, E., and Barfett, J. (2018). Generalization of deep neural networks for chest pathology classification in X-rays using generative adversarial networks. In *ICASSP*. 22, 24

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *NIPS*. 23, 40, 73

Shiraishi, J., Katsuragawa, S., et al. (2000). Development of a digital image database for chest radiographs with and without a lung nodule. *J of Roent*. 113

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. 11

Son, J., Park, S. J., and Jung, K.-H. (2017). Retinal vessel segmentation in fundoscopic images with generative adversarial networks. *arXiv preprint arXiv:1706.09318*. 84

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*. 24

Sun, H., Zhen, X., Bailey, C., Rasoulinejad, P., Yin, Y., and Li, S. (2017). Direct estimation of spinal cobb angles by structured multi-output regression. In *International Conference on Information Processing in Medical Imaging*, pages 529–540. Springer. 20

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9. 11

Taghizadeh, E., Terrier, A., Becce, F., Farron, A., and Büchler, P. (2019). Automated ct bone segmentation using statistical shape modelling and local template matching. *Computer Methods in Biomechanics and Biomedical Engineering*, 22(16):1303–1310. 20

Tajbakhsh, N., Hu, Y., Cao, J., Yan, X., Xiao, Y., Lu, Y., Liang, J., Terzopoulos, D., and Ding, X. (2019a). Surrogate supervision for medical image analysis: Effective deep learning from limited quantities of labeled data. *arXiv preprint arXiv:1901.08707*. 21

Tajbakhsh, N., Lai, B., Ananth, S., and Ding, X. (2019b). Errornet: Learning error representations from limited data to improve vascular segmentation. *arXiv preprint arXiv:1910.04814*. 25, 84

Tran, P. V. (2019). Semi-supervised learning with self-supervised networks. *arXiv preprint arXiv:1906.10343*. 21

Unterthiner, T., Nessler, B., Seward, C., Klambauer, G., Heusel, M., Ramsauer, H., and Hochreiter, S. (2017). Coulomb gans: Provably optimal nash equilibria via potential fields. *arXiv preprint arXiv:1708.08819*. 74

van Rikxoort, E., Prokop, M., de Hoop, B., Viergever, M., Pluim, J., and van Ginneken, B. (2010). Automatic segmentation of pulmonary lobes robust against incomplete fissures. *IEEE Tran on Med Imag*, 29(6):1286–1296. 17, 63

Wang, S. and Zhang, L. (2017). Catgan: Coupled adversarial transfer for domain generation. *arXiv preprint arXiv:1711.08904*. 24

Wang, X., Teng, P., Lo, P., Banola, A., Kim, G., Abtin, F., Goldin, J., and Brown, M. (2018). High throughput lung and lobar segmentation by 2D and 3D CNN on chest CT with diffuse lung disease. In *Image Analysis for Moving Organ, Breast, and Thoracic Images*, pages 202–214. Springer. 17

Weinstein, S. L., Dolan, L. A., Cheng, J. C., Danielsson, A., and Morcuende, J. A. (2008). Adolescent idiopathic scoliosis. *The Lancet*, 371(9623):1527–1537. 18

Wong, C. C. and McGirt, M. J. (2013). Vertebral compression fractures: a review of current management and multimodal therapy. *Journal of multidisciplinary healthcare*, 6:205. 114

Wu, H., Bailey, C., Rasoulinejad, P., and Li, S. (2017). Automatic landmark estimation for adolescent idiopathic scoliosis assessment using boostnet. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 127–135. Springer. 20

Wu, H., Bailey, C., Rasoulinejad, P., and Li, S. (2018). Automated comprehensive adolescent idiopathic scoliosis assessment using MVC-Net. *Medical image analysis*, 48:1–11. 20

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*. 99

Yang, J., Dvornek, N. C., Zhang, F., Zhuang, J., Chapiro, J., Lin, M., and Duncan, J. S. (2019). Domain-agnostic learning with anatomy-consistent embedding for cross-modality liver segmentation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 25

Yang, S., Andras, L. M., Redding, G. J., and Skaggs, D. L. (2016). Early-onset scoliosis: a review of history, current treatment, and future directions. *Pediatrics*, 137(1):e20150709. 33

Yang, X., Zeng, Z., Yeo, S. Y., et al. (2017). A novel multi-task deep learning model for skin lesion segmentation and classification. *arXiv:1703.01025*. 26

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. (2019). S^4L: Self-supervised semi-supervised learning. *arXiv preprint arXiv:1905.03670*. 21

Zhang, Y., Wei, Y., and Yang, Q. (2018a). Learning to multitask. In *Advances in Neural Information Processing Systems*, pages 5771–5782. 26

Zhang, Z., Yang, L., and Zheng, Y. (2018b). Translating and segmenting multimodal medical volumes with cycle-and shape-consistency generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern Recognition*, pages 9242–9251. 25

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. (2019). Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE Transactions on Medical Imaging*. 12

Zhu, W., Huang, Y., , et al. (2019). Anatomynet: Deep learning for fast and fully automated whole-volume segmentation of head and neck. *Med Phys*. 95

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2019). A comprehensive survey on transfer learning. *arXiv preprint arXiv:1911.02685*. 25